Universidad de Buenos Aires
Facultad de Ingeniería

75.99 Trabajo Profesional

Sistema Experto para la Identificación de Riesgos en el Desarrollo de Software:

P.L. Risk Identification System (RIS)

Director:

Dr. Ramón García-Martínez

Co - Directores:

M. Ing. Hernán Merlino
M. Ing. Enrique Fernández

Alumnos:

Nicolás Ledo 81192
Daniel Palacios 80380

Resumen

Tanto en el ámbito laboral como el académico una de las principales dificultades que se encuentran a la hora de desarrollar una aplicación informática es la identificación de riesgos. Es por éste motivo que nace P.L. Risk Identification System.
P.L. tuvo como objetivo primordial ofrecerle al usuario una vía rapida, eficiente e intuitiva de poder detectar, informar e individualizar riesgos en el proceso de desarrollo de software mediante una serie preguntas.
Resulta ser una poderosa herramienta, portable (por ser una aplicacion web) y muy sencilla de utilizar.

Abstract

Both in the working and the academic environments one of the main difficulties encountered when developing a computer application is the identification of risks.
It is for this reason P.L. Risk Identification System was developed.
P.L. Risk Identification System's primary objective was to offer the user a rapid, efficient and intuitive tool to detect, report and identify risks in the software development process through a series of questions.
It turns out to be a powerful technology, which is not only portable (it's in fact a web application) buy very easy to use.

Índice

Resumen 2
Abstract 2
1. Introducción 6
2. Visión 6
3. Dominio de Aplicación 6
3.1 Sistema Experto 8
4. Estado del Arte 10
5. Metodología IDEAL 12
5.1 Fase I - Identificación de la tarea 12
5.2 Fase II - Desarrollo de los prototipos 13
5.3 Fase III - Ejecución de la construcción del sistema integrado 15
5.4 Fase IV - Actuación para conseguir el mantenimiento perfectivo 15
5.5 Fase V - Lograr una adecuada transferencia tecnológica 16
6. Estimación de Tiempos y Diagramas de Gantt 17
6.1 Estimación de Tareas 17
6.2 Estimación de Sub-Tareas 18
7. Descripción de Sub-tareas a realizar 21
7.1 Fase 1 - Identificación de la tarea 21
7.2 Fase 2 - Desarrollo de los distintos prototipos 21
7.3 Fase 3 - Ejecución de la construcción del sistema integrado 22
7.4 Fase 4 - Actuación para conseguir el mantenimiento perfectivo 23
7.5 Fase 5 - Lograr una adecuada transferencia tecnológica 23
8. Estudio de Viabilidad 24
8.1 Dimensión de Plausivilidad 24
8.2 Dimensión de Justificación 25
8.3 Dimensión de Adecuación 27
8.4 Dimensión de Éxito 30
8.5 Resultados 35
9. Análisis Estructural de textos 36
9.1 Tabla N° 1 36
10. Cálculo del Emparrillado 50
10.1 Clasificación de los Elementos 51
10.1.1 Árbol de Elementos 52

10.1.2 Análisis de los resultados 52
10.2 Clasificación de Características 52
10.2.1 Árbol de Características 55
10.2.2 Análisis de los resultados 55
11. Conocimientos Fácticos 55
11.1 Glosario de Términos 56
11.1.1 Tabla N° 2 56
11.1.2 Anexo (Tabla N° 2) 70
11.2 Tabla Concepto – Atributo – Valor (TCAV) 73
11.2.1 Tabla N° 3 (TCAV) 73
11.2.2 Anexo Tabla N° 3 (TCAV) 77
11.3 Mapa de relaciones 82
12. Conocimiento Estratégico 83
12.1 Árbol de Descomposición Funcional 84
12.2 Descripción de Estrategias 84
13. Conocimiento Táctico 89
13.1 Tablas PER (Palabras del Experto-Regla) 89
13.2 Reglas para conclusiones 106
14. Módelo Dinámico 117
14.1 Introducción 117
14.2 Mapa de Conocimientos 117
14.3 Árbol Jerárquico 119
15. Formalización 124
15.1 Introducción 124
15.2 Marcos 124
15.2.1 Marco clase PL RIS 124
15.2.2 Marco clase Requirements 124
15.2.3 Marco clase Design 125
15.2.4 Marco clase Code and Unit Test 126
15.2.5 Marco clase Integration and Test 126
15.2.6 Marco clase Engineering Specialities 127
15.2.7 Marco clase Development Process 127
15.2.8 Marco clase Development System 128
15.2.9 Marco clase Management Process 129
15.2.10 Marco clase Management Methods 129

15.2.11 Marco clase Work Environment 130
15.2.12 Marco clase Resources 130
15.2.13 Marco clase Contract 131
15.2.14 Marco clase Program Interfaces 131
5.3 Reglas de Producción 132
16. Referencias 133
17. Anexo I. Manual de Usabilidad 135
17.1 Introducción 135
17.2 SE – Sistema Experto 136
17.2.1 Motor de Inferencia 136
17.2.2 Reglas 136
18. Sistema Experto para la Identificación de Riesgos en el Desarrollo de Software 142
18.1 Análisis y diseño 142
18.1.1 Introducción 142
18.1.2 Arquitectura del sistema 143
18.2 Configuración 144
18.2.1 Requerimientos de Hardware 144
18.2.2 Requerimientos de Software 144
18.2.3 Ejecución – Modo de uso 144
19. Anexo II. Procesador de lenguaje natural 149
19.1 Introduccion - NLP (Natural Language Processing) 149
19.2 Soluciones e implementación propuesta 149
20. Anexo III. Herramientas utilizadas y aclaraciones 152
21. Anexo IV. Usability Manual 153
21.1 Configuration 153
22.1.1 Hardware Requirements 153
22.1.2 Software Requirements 153
21.2 Execution 153

1. Introducción

Este proyecto consiste en la realización de un Sistema Experto basado en el informe Taxonomy-Based Risk Identification realizado por el Software Engineering Institute (SEI) de la Universidad Carnegie Mellon.

El mismo consiste en un método sistemático y repetible que facilita la identificación de riesgos en proyectos de desarrollo de software. Para tal fin, existe un cuestionario basado en taxonomías (Taxonomy-Base questionnaire, TBQ) que organiza el proceso de desarrollo en tres niveles: clases, elementos y atributos. El cuestionario tiene preguntas para cada uno de estos niveles de manera de no dejar ningún aspecto sin revisar.

El SE en desarrollo emulará entonces este cuestionario para así automatizar el uso de esta técnica.

2. Visión

Desarrollar un Sistema Experto basado en el informe Taxonomy-Based Risk Identification realizado por el Software Engineering Institute (SEI) de la Universidad Carnegie Mellon.

El mismo consistirá en un método sistemático y repetible que facilitará la identificación de riesgos en proyectos de desarrollo de software.

3. Dominio de Aplicación

Para poder determinar el dominio de la aplicación debemos tener una idea bien concreta del problema que se intenta solucionar.
Hoy en día lograr minimizar los riesgos en el desarrollo de software es una pieza clave para mejorar características como:

- Calidad
- Confiabilidad
- Estabilidad
- Escalabilidad

- Funcionalidad
- Performance
- Mantenimiento
- Seguridad
- Capacidad
- Usabilidad
- Testeo
- Codificación/Implementación
- Presupuesto
- Recursos humanos
- Etc.

Es sabido que existen muy pocas personas idóneas en el tema de la detección de riesgos en el desarrollo e implementación de software. Muchas empresas y particulares realizan desarrollos de software sin reparar demasiado en todos los riesgos que un desarrollo tiene asociado, lo cual influye negativamente tanto para la empresa (demandará más tiempo del estipulado el desarrollo del sistema, lo cual se traduce directamente en una reducción del ingreso por el sistema) como para el cliente (tardará más en tener el sistema funcionando). Es por eso que consideramos que es de vital importancia poder llevar los conocimientos de un experto en el ramo a un sistema que pueda ser utilizado por la comunidad de la informática.

Por lo general es muy difícil contar con la ayuda de los expertos y cuando se logra poder contar con ella es necesario poder volcar esa información a un sistema, ya que este puede ser utilizado en cualquier momento que se desee. Es en este ámbito justamente donde se aplican los Sistemas Expertos. Lo que diferencia a estos sistemas de uno tradicional de recuperación de información es que éstos últimos sólo son capaces de recuperar lo que existe explícitamente, mientras que un Sistema Experto debe ser capaz de generar información no explícita, razonando con los elementos que se le dan. Pero la capacidad de los SE en el ámbito de la recuperación de la información no se limita a la simplemente a la recuperación. Pueden utilizarse para ayudar al usuario, en selección de recursos de información, en filtrado de respuestas, etc. Un SE puede actuar como un intermediario inteligente que guía y apoya el trabajo del usuario final.

Es necesario explicar entonces cuales son las características y los problemas que logra resolver en general un sistema experto.

A continuación pasamos a desarrollar este tema.

3.1 Sistema experto

Los sistemas expertos son llamados así porque emulan el comportamiento de un experto en un dominio concreto y en ocasiones son usados por éstos. Con los sistemas expertos se busca una mejor calidad y rapidez en las respuestas dando así lugar a una mejora de la productividad del experto.

Se puede entender como una rama de la inteligencia artificial. Estos sistemas imitan las actividades de un humano para resolver problemas de distinta índole (no necesariamente tiene que ser de inteligencia artificial). También se dice que un SE se basa en el conocimiento declarativo (hechos sobre objetos, situaciones) y el conocimiento de control (información sobre el seguimiento de una acción).
Para que un sistema experto sea una herramienta efectiva, los usuarios deben interactuar de una forma fácil, reuniendo dos capacidades para poder cumplirlo:

Explicar sus razonamientos o base del conocimiento: los sistemas expertos se deben realizar siguiendo ciertas reglas o pasos comprensibles de manera que se pueda generar la explicación para cada una de estas reglas, que a la vez se basan en hechos.

Adquisición de nuevos conocimientos o integrador del sistema: son mecanismos de razonamiento que sirven para modificar los conocimientos anteriores. Sobre la base de lo anterior se puede decir que los sistemas expertos son el producto de investigaciones en el campo de la inteligencia artificial ya que ésta no intenta sustituir a los expertos humanos, sino que se desea ayudarlos a realizar con más rapidez y eficacia todas las tareas que realiza.

Debido a esto en la actualidad se están mezclando diferentes técnicas o aplicaciones aprovechando las ventajas que cada una de estas ofrece para poder tener empresas más seguras. Un ejemplo de estas técnicas sería los agentes que tienen la capacidad de negociar y navegar a través de recursos en línea; y es por eso que en la actualidad juega un papel preponderante en los sistemas expertos.

Un Sistema Experto está conformado por:

- Base de conocimientos (BC): Contiene conocimiento modelado extraído del diálogo con el experto.
- Base de hechos (Memoria de trabajo): contiene los hechos sobre un problema que se ha descubierto durante el análisis.
- Motor de inferencia: Modela el proceso de razonamiento humano.

- Módulos de justificación: Explica el razonamiento utilizado por el sistema para llegar a una determinada conclusión.
- Interfaz de usuario: es la interacción entre el SE y el usuario, y se realiza mediante el lenguaje natural.

Ventajas

- Permanencia: A diferencia de un experto humano un SE (sistema experto) no envejece, y por tanto no sufre pérdida de facultades con el paso del tiempo.
- Duplicación: Una vez programado un SE lo podemos duplicar infinidad de veces.
- Rapidez: Un SE puede obtener información de una base de datos y realizar cálculos numéricos mucho más rápido que cualquier ser humano.
- Bajo costo: A pesar de que el costo inicial pueda ser elevado, gracias a la capacidad de duplicación el coste finalmente es bajo.
- Entornos peligrosos: Un SE puede trabajar en entornos peligrosos o dañinos para el ser humano.
- Fiabilidad: Los SE no se ven afectados por condiciones externas, un humano sí (cansancio, presión, etc.).
- Consolidar varios conocimientos
- Apoyo Académico.
- Etc.

Limitaciones

- Sentido común: Para un Sistema Experto no hay nada obvio. Por ejemplo, un sistema experto sobre medicina podría admitir que un hombre lleva 40 meses embarazado, a no ser que se especifique que esto no es posible.
- Lenguaje natural: Con un experto humano podemos mantener una conversación informal mientras que con un SE no podemos.
- Capacidad de aprendizaje: Cualquier persona aprende con relativa facilidad de sus errores y de errores ajenos, que un SE haga esto es muy complicado.
- Perspectiva global: Un experto humano es capaz de distinguir cuales son las cuestiones relevantes de un problema y separarlas de cuestiones secundarias.
- Capacidad sensorial: Un SE carece de sentidos.

- Flexibilidad: Un humano es sumamente flexible a la hora de aceptar datos para la resolución de un problema.
- Conocimiento no estructurado: Un SE no es capaz de manejar conocimiento poco estructurado.

Actualmente el difícil y cambiante mercado competitivo se vuelve más complejo por la gran diversidad de información que se ven obligados a almacenar y analizar, razón por la cual las empresas se ven en la necesidad de recurrir a poderosas y/o robustas herramientas o sistemas que les sirvan de soporte a la hora de tomar decisiones. De esta forma estos inteligentes, precisos y eficientes sistemas son adoptados por más organizaciones, en las cuales se convierten y/o transforman en una importante estrategia de negocio.
Por otra parte es importante mencionar que estos seguirán siendo usados en todas y cada una de las áreas y/o campos donde los expertos humanos sean escasos. Por consecuencia de lo anterior estos sistemas son utilizados por personas no especializadas, por lo cual el uso frecuente de los (SE) les produce y/o genera conocimiento a los usuarios.

4. Estado del Arte

Si bien no se encuentran en el mercado sistemas que intenten resolver el problema abordado en el informe utilizando sistemas expertos, se han encontrado diversos sistemas expertos para la detección de riesgos y fallas aplicados en distintos dominios.
Se describirán algunos de éstos sistemas a continuación.

Sistema Experto de Asistencia Técnica de Automóviles
Este sistema se implementó por medio del lenguaje de programación lógica Turbo Prolog 2.0 y pertenece a la categoría de "Sistemas Expertos de Diagnostico".

Descripción del proceso de detección de fallas.
Cuando un vehículo es abordado por el técnico, este debe separar y desmantelar la partes y piezas de dicho auto. Una vez terminado se realiza un análisis para detectar las posibles fallas o defectos. Estos defectos se agrupan en conjuntos a los cuales se les denomina Problemas. Una vez detectado el problema se puede buscar una posible solución.

Es importante notar que pueden existir defectos que pertenezcan a distintos problemas; es decir, que a diferentes problemas se le asignan uno o más defectos iguales.
Es así que un grupo de defectos también puede desembocar en más de un problema; es por esto que el sistema experto da consejos de cómo solucionar los posibles problemas, no la solución óptima.
Una vez tenidos los problemas presentados por el o los conjuntos de defectos, se procede a desplegar o informar sobre las soluciones aconsejables.

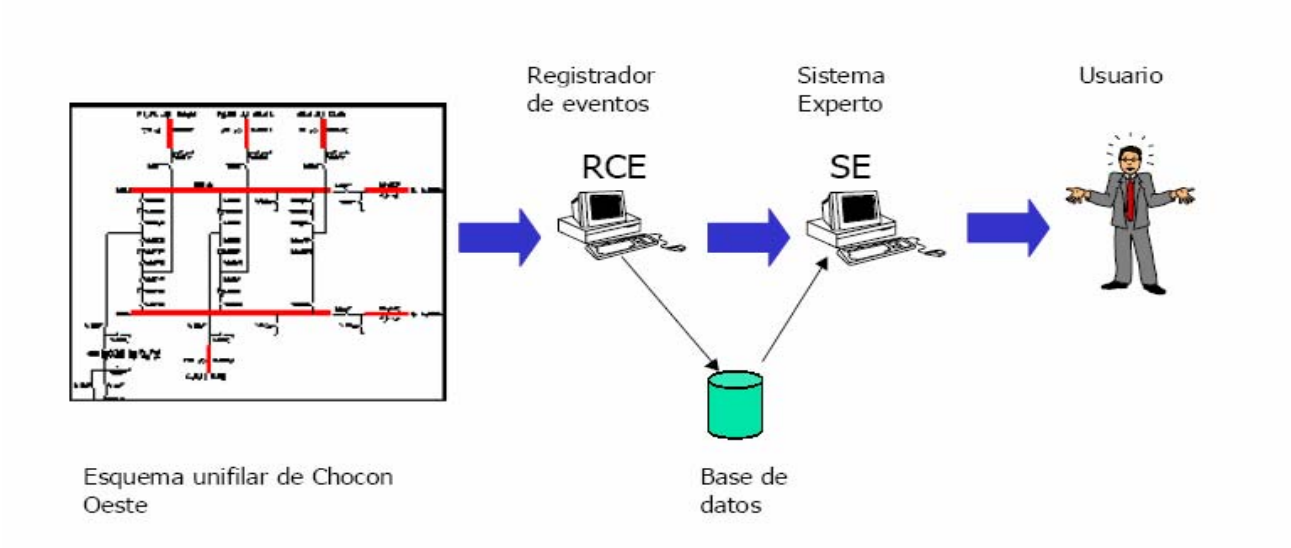
Sistema Experto en Análisis de Fallas en Líneas Eléctricas de Transmisión

Los Sistemas eléctricos de Transmisión están sometidos a diversos fenómenos (contingencias) que producen distintos tipos de fallas (perturbaciones) eléctricas. Entre los fenómenos físicos causantes de una falla eléctrica se puede mencionar: viento, incendio de campo, la caída de una torre, maniobras, descargas atmosféricas, etc. Estos fenómenos pueden originar diversos tipos de fallas.

Descripción del proceso de detección de fallas.

El Sistema Experto procesa en tiempo real la información adquirida por el Registrador de Eventos y frente a un suceso característico de una falla, emite un diagnóstico previo que asistirá a los especialistas y los operadores a identificar rápidamente el origen del problema y efectuar las operaciones que correspondan.

En el siguiente gráfico se puede observar un esquema de la solución propuesta



A la izquierda de la figura se puede observar el unifilar de una estación, los eventos generados en esta son almacenado en una base de datos por el Registrador Cronológico de Eventos (RCE), anta la ocurrencia de una falla, el RCE, habilita al SE. Este sistema lee los eventos registrados y emite un

diagnóstico que permitirá a los operadores revisar el origen posible de la falla y facilitará a los especialistas la información del SE para el análisis de la situación.

Al igual que en los dos casos detallados anteriormente, los sistemas expertos se pueden utilizar de manera satisfactoria para la detección de riesgos en el desarrollo de sistemas de software. Es por este motivo que el presente proyecto proporciona una herramienta de apoyo de gran relevancia para detectar riesgos en el desarrollo de software, constituyéndose en una importante ayuda para los desarrolladores del mismo. Esta herramienta tendrá la posibilidad de almacenar gran cantidad de información y realizar un gran número de operaciones en poco tiempo de manera que se obtengan conclusiones rápidamente.

Con las investigaciones realizadas durante la realización del proyecto pudimos concluir que a pesar de existir un sin número de sistemas expertos para resolver distintas problemáticas, ninguno aborda puntualmente el problema de la identificación de riesgos en el desarrollo de software. Es por eso que no sólo es una herramienta novedosa sino que da soporte y soluciones a un problema muy común y de gran importancia para los tiempos que corren.

5. Metodología IDEAL

La metodología IDEAL consta de cinco fases, a saber:

1. Identificación de la tarea.
2. Desarrollo de los prototipos.
3. Ejecución de la construcción del sistema integrado.
4. Actuación para conseguir el mantenimiento perfectivo.
5. Lograr una adecuada transferencia tecnológica.

Cada una de éstas fases se “subdivide” en distintas etapas, las cuales se explicarán a continuación.

5.1 Fase I. Identificación de la tarea

Esta fase considera la definición de los objetivos de la aplicación y, en base a ellos, determinar si la tarea es susceptible de ser tratada con la tecnología de la Ingeniería del Conocimiento (en adelante INCO). En caso afirmativo, se definen las características del problema y se especifican los

requisitos que enmarcarán la solución del problema. Para ello, esta fase se divide en las tres etapas siguientes:

Etapa 1.1. Plan de requisitos y adquisición de conocimientos.

Se identifican las necesidades del cliente describiendo cuales son los objetivos del sistema, qué informaciones se van a obtener y suministrar, funcionalidades a exigir y requisitos necesarios para alcanzar todo ello. Para confeccionar el plan de requisitos es necesario comenzar con la adquisición de conocimientos, entrevistándose con directivos, expertos y usuarios. La adquisición profunda se llevará a cabo en la fase II.

Etapa I.1. Evaluación y selección de la tarea.

Esta etapa conforma el estudio de viabilidad, desde la perspectiva de la INCO, cuantificando dicha evaluación para ver qué grado de dificultad presenta la tarea. Esta etapa es fundamental para evitar a priori fallos detectados en la aplicación práctica de esta tecnología.

Etapa I.3. Definiciones de las características de la tarea.

Aquí, se establecen las características más relevantes asociadas con el desarrollo de la aplicación. Una definición de la aplicación desde el punto de vista del sistema. Es decir, una especificación técnica completa emitida por el Ingeniero del Conocimiento (en adelante IC). Se debe llevar a cabo una especificación inicial de los siguientes tipos de requisitos: funcionales, operativos, de interfaz, de soporte, criterios de éxito, casos de prueba o juego de ensayo. Recursos materiales y humanos para desarrollar el Sistema Experto (en adelante SE). Análisis de costes/beneficios y evaluación de riesgos. Hitos y calendario. En esta fase los expertos, usuarios y directivos, consiguen perfilar el ámbito del problema; definir funcionalidades, rendimiento, e interfaces; analizar el entorno de la tarea y del riesgo de desarrollo del SE. Todo ello hace que el proyecto se justifique, y asegura que los IICC y los clientes tengan la misma percepción de los objetivos del sistema.

5.2 Fase II. Desarrollo de los distintos prototipos

Esta fase concierne al desarrollo de prototipos que permiten definir y refinar las especificaciones del sistema. A continuación se describen los prototipos de: investigación, campo y operación, que son sucesivos refinamientos cada uno del anterior.

Etapa II.1. Concepción de la solución.

Produce un diseño general del sistema prototipo. El IC y el experto estudian las especificaciones parciales del sistema y el plan del proyecto y, en base a ellos, producen un diseño general.

Etapa II.2. Adquisición y conceptualización de los conocimientos.

La adquisición, tanto en la extracción de los conocimientos públicos (libros, documentos, manuales de procedimientos, etc.) como en la educción de los conocimientos privados de los expertos, se alterna con la conceptualización para modelar el comportamiento del experto. La conceptualización permite entender el dominio del problema a partir de la información obtenida en la etapa de adquisición.

Etapa II.3. Formalización de los conocimientos.

Se seleccionan los formalismos para representar los conocimientos que conforman la conceptualización obtenida, y el diseño detallado del SE. Este último es en una estructura modular del sistema que incorpora los conceptos que participan en el prototipo. Se establecen los módulos que definen el motor de inferencias, la base de conocimientos, interfaces (de usuario y a otros sistemas), etc.

Etapa II.4. Implementación.

Si en la etapa anterior se seleccionó una herramienta de desarrollo adecuada y el problema se ajusta a ella y viceversa, la implementación es inmediata y automática. En otro caso, es necesario programar, al menos, parte del Sistema Basado en Conocimiento (en adelante SBC).

Etapa II.5. Validación y evaluación.

La fiabilidad es el punto más sensible de todo SE y por tanto su punto crítico dado que estos sistemas están construidos para contextos en los que las decisiones son, en gran medida, discutibles. Sin embargo, existen técnicas que permiten realizar esta validación de una forma razonablemente satisfactoria. Para ello, se deben realizar las siguientes
acciones. Casos de prueba o juego de ensayo que, a modo de Test de Turing,
permiten comparar las respuestas de los expertos frente a las del sistema y ver si hay discrepancias o no. Ensayo en paralelo que es una consecuencia del anterior y consiste en que los expertos usen rutinariamente el SE desarrollado para ver las discrepancias entre ambos.

Etapa II.6. Definición de nuevos requisitos, especificaciones y diseño.

Los SSBBCC se construyen de forma incremental, generando primero un prototipo de investigación, que se convierte en un prototipo de campo para, finalmente, resultar un prototipo de operación. Esta etapa se corresponde con la definición de los requisitos, especificaciones y diseño

del siguiente prototipo, que para ser construido deberá pasarse, de nuevo, por las etapas II.1 a II.5. Esta fase acaba con la obtención del
sistema experto completo.

Las etapas 2 a 6 se repiten para cada prototipo.

5.3 Fase III. Ejecución de la construcción del sistema integrado

La fase III consta de:

Etapa III.1. Requisitos y diseño de la integración con otros sistemas.
Es el estudio y diseño de interfaces y puentes con otros sistemas hardware y software.

Etapa III.2. Implementación y evaluación de la integración.
Su fin es desarrollar, utilizando técnicas de IS, los requisitos de la etapa anterior. Esto es, esta etapa implemento la integración del SE con los otros sistemas hardware y software, para conseguir un sistema final.

Etapa III.3. Aceptación por el usuario del sistema final.
Es la prueba última de aceptación por los expertos y usuarios finales, que debe satisfacer todas sus expectativas y exigencias, tanto en lo concerniente a su fiabilidad como eficiencia.

Fase IV. Actuación para conseguir el mantenimiento perfectivo

Trata del mantenimiento del sistema, dadas las características específicas de los SSBBCC, el mantenimiento perfectivo es esencial, puesto que, además del aumento de funcionalidades, efectúa la incorporación de nuevos conocimientos que, sin duda, se van a generar por el propio uso del SBC. En este el análisis de protocolos, como forma de adquisición de conocimientos, es imprescindible.

Etapa IV.l Definir el mantenimiento del sistema global.
Esta etapa emplea las técnicas de IS, definiendo el mantenimiento que se llevará a cabo igual que en cualquier otro tipo de sistema.

Etapa IV.2. Definir el mantenimiento de las bases de conocimientos.
Existen diversas técnicas para el mantenimiento de bases de conocimiento.

Etapa IV.3. Adquisición de nuevos conocimientos.

Diseñar protocolos para que cuando aparezcan nuevos conocimientos, puedan captarse y registrarse. Se deben establecer métodos para actualizar el sistema incorporando los conocimientos adquiridos.

Fase V. Lograr una adecuada transferencia tecnológica

Se encarga de la transferencia tecnológica. Cualquier sistema necesita, para su correcta implantación y uso rutinario, una adecuada transferencia de manejo. No resulta lo mismo cuando el sistema es usado por sus constructores que por los usuarios del mismo. El único modo de eliminar estas diferencias es mediante una meticulosa transferencia tecnológica, que engloba las dos etapas siguientes:

Etapa V.l. Organizar la transferencia tecnológica

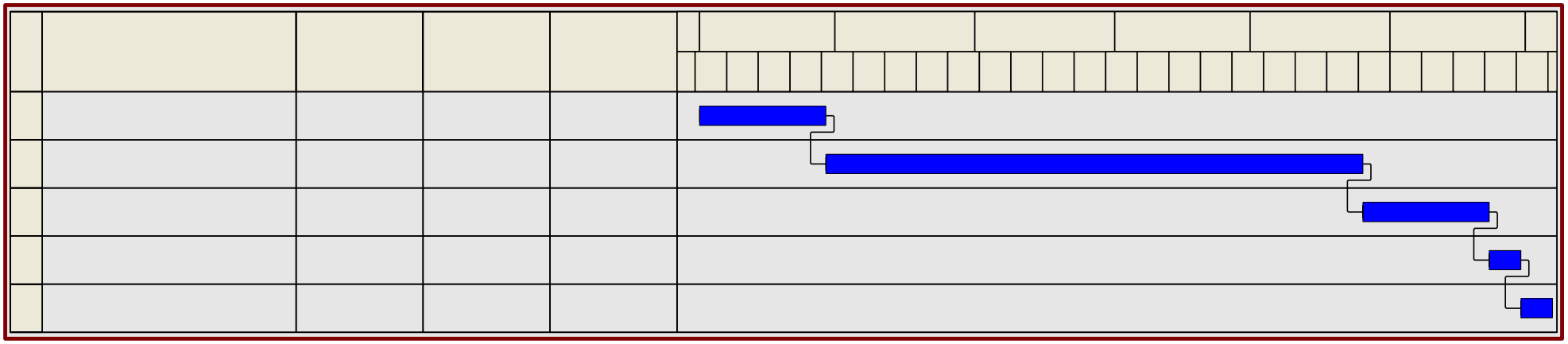
Meticulosamente mediante entrenamiento en sesiones de tutoría entre los diseñadores y los usuarios que sirvan tanto para explicar el manejo del propio sistema como para manejar y entender la documentación del mismo.

Etapa V.2. Completar la documentación del sistema

desde el dossier técnico al manual del usuario, que deben incorporar todas las peculiaridades de su uso de una forma amigable para el usuario final a quien debe ir dirigido.

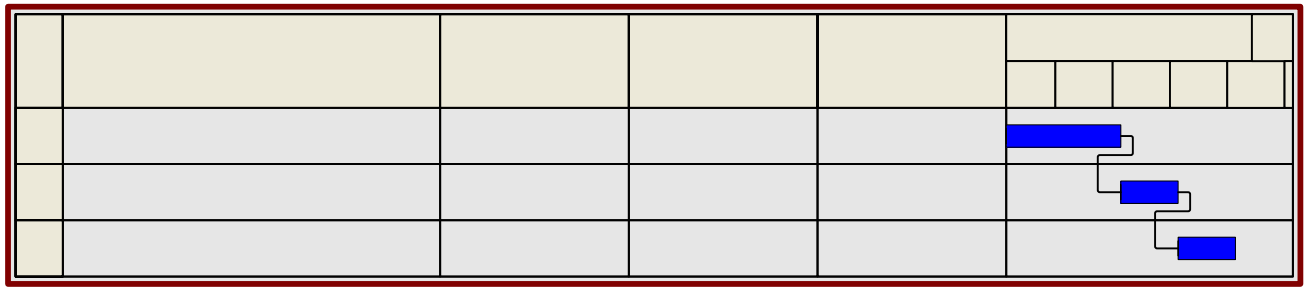
6. Estimación de Tiempos y Diagramas de Gantt

6.1 Estimación de Tareas



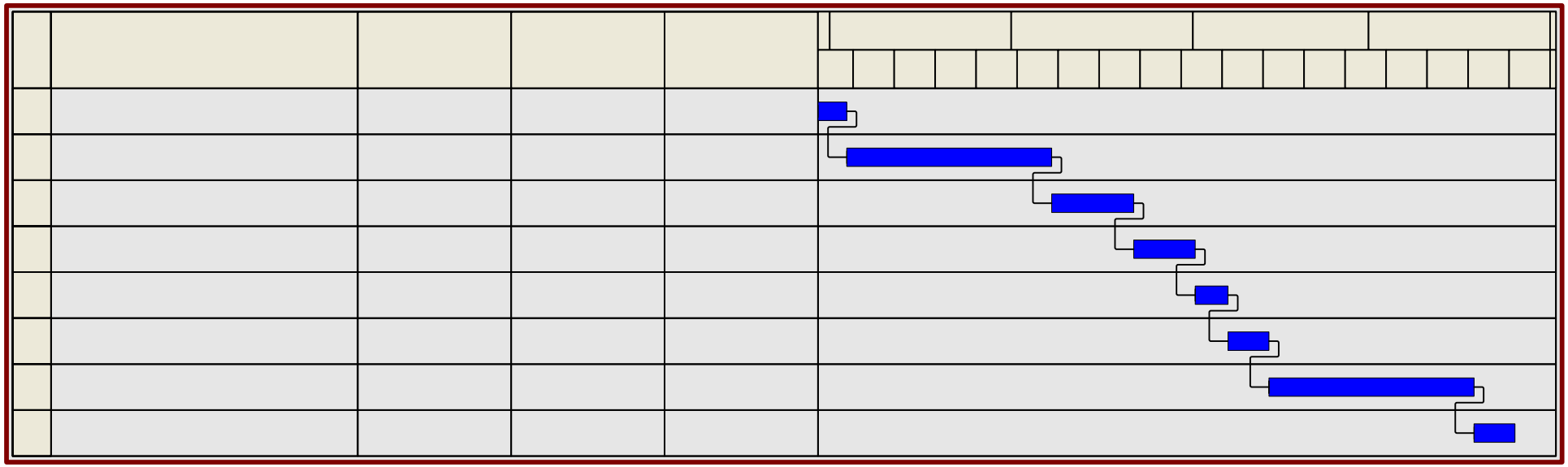
6.2 Estimación de Sub-Tareas

1. Identificación de la tarea

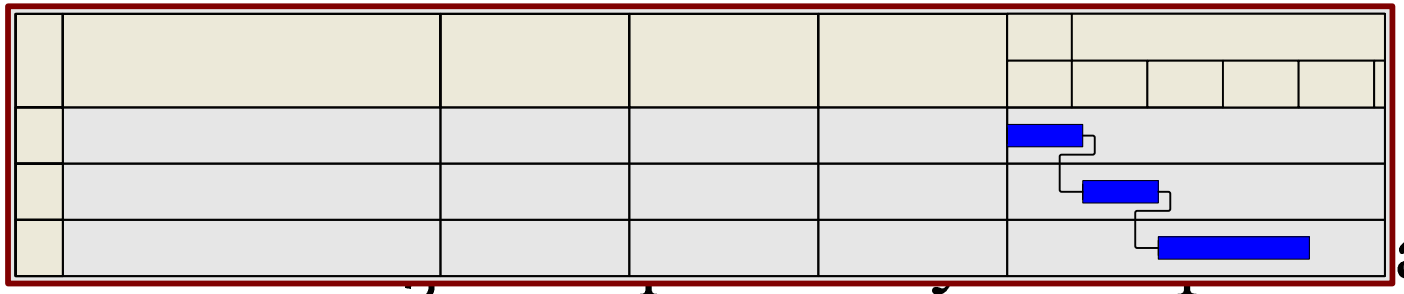


Id. Nombre de tarea Comienzo

2. Desarrollo de los distintos prototipos

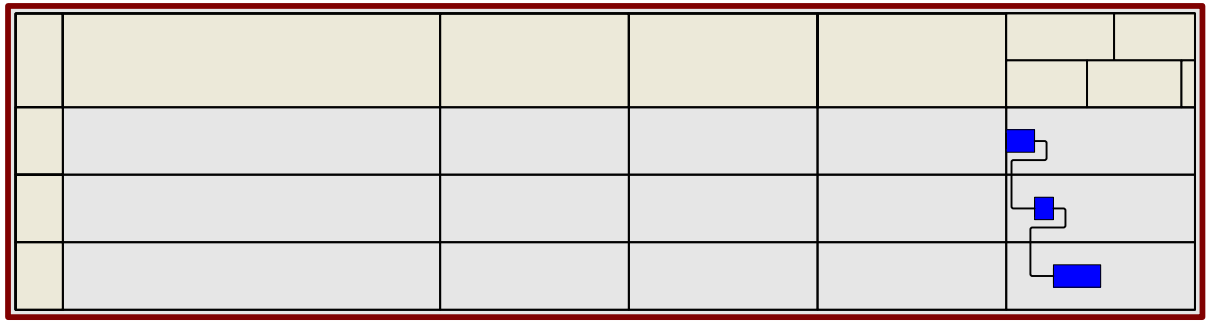


3. Ejecución de la construcción del sistema integrado

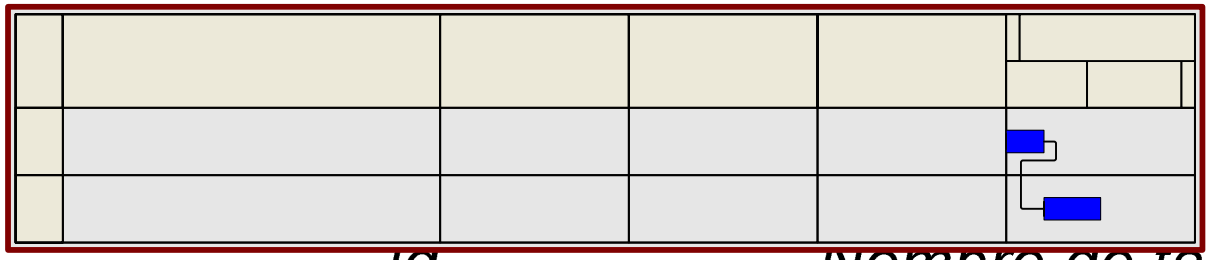


Id.	Nombre de tarea	Comienzo
2	ación de los conocimientos	29/06/2009
3	Formalización de los	03/07/2009
		07/08/2009

4. Actuación para conseguir el mantenimiento perfectivo



5. Lograr una adecuada transferencia tecnológica



Id.	Nombre de tarea	Comienzo
1	Definir el mantenimiento del sistema global	23/11/2009
2	Definir el mantenimiento de las	25/11/2009

7. Descripción de las Sub-tareas a realizar

7.1 Fase 1 - Identificación de la tarea

Plan de requisitos y adquisición de conocimientos

Lectura y análisis del informe *Taxonomy-Based Risk Identification* realizado por el *Software Engineering Institute (SEI)* de la Universidad Carnegie Mellon. Entrevista a Expertos y Usuarios

Evaluación y selección de la tarea

Método de "estudio de viabilidad"

Definiciones de las características de la tarea

Análisis de los requisitos: funcionales, operativos, de interfaz, de soporte, criterios de éxito, casos de prueba o juego de ensayo

7.2 Fase 2 - Desarrollo de los distintos prototipos

Concepción de la solución

- Análisis de la metodología a utilizar
- Selección de la metodología: Metodología Ideal

Adquisición y conceptualización de los conocimientos

- Método de "Análisis estructural de texto"
- Método de "Emparrillado"
- Conocimientos Fácticos. Tabla Concepto-Atributo-valor
- Mapa de Relaciones
- Conocimiento Estratégico. Árbol de Descomposición funcional
- Conocimiento Estratégico. Descripción de Estrategias
- Conocimiento Táctico. Tablas PER
- Conocimiento Táctico. Seudo reglas
- Modelo Dinámico. Mapa de Conocimientos
- Modelo Dinámico. Árbol Jerárquico

Formalización de los conocimientos

- Se establecen los módulos que definen el motor de inferencias, la base de conocimientos, interfaces (de usuario y a otros sistemas)
- Formalización. Marcos

Análisis y selección de herramientas a utilizar

- Análisis de las herramientas a utilizar
- Selección de herramientas a utilizar: clips para motor de inferencia. JSP, Struts2, HTML, etc. para el proyecto Web global

Selección de la arquitectura del sistema

- Análisis de las distintas arquitecturas posibles y selección de la que mejor se adapta al negocio
- Selección de la arquitectura: El Modelo Vista Controlador (MVC)

Implementación de casos de prueba

Desarrollo de casos de prueba para el futuro testeo de la aplicación, realizados en base a las necesidades del cliente y el modelo del negocio.

Implementación de la aplicación

- Implementación del motor de inferencia y del proyecto Web teniendo en cuenta la arquitectura seleccionada.
- Desarrollo de la lógica de Negocio
- Desarrollo de la Vista (HTML)

Integración

Creación de un Wrapper para la comunicación entre el proyecto Web realizado en Java y el motor de inferencia realizado en CLIPS

7.3 Fase 3 - Ejecución de la construcción del sistema integrado

Testing y Análisis correctivo

Testeo de la aplicación integrada con los casos de prueba propuestos y corrección de los bugs.

Lanzamiento de Versión Beta

Testing de la aplicación por parte del cliente

Puesta a punto de la aplicación

Corrección de bugs y lanzamiento de versión definitiva

7.4 Fase 4 - Actuación para conseguir el mantenimiento perfectivo

Definir el mantenimiento del sistema global

Definir el mantenimiento que se llevará a cabo igual que en cualquier otro tipo de sistema

Definir el mantenimiento de las bases de conocimientos

Definir como se hará el mantenimiento

Adquisición de nuevos conocimientos

Diseñar protocolos para que cuando aparezcan nuevos conocimientos, puedan captarse y registrarse

7.5 Fase 5 - Lograr una adecuada transferencia tecnológica

Organizar la transferencia tecnológica

Planificar las sesiones de tutoría entre los diseñadores y
los usuarios

Completar la documentación del sistema

Desarrollo del manual del usuario

8. Estudio de Viabilidad

El desarrollo del sistema experto debe estar justificado de alguna manera que nos permita estimar si el mismo será apropiado y si será exitoso. Para ello, se realizó el Cálculo de Viabilidad, en el que se analizan características para cuatro dimensiones del proyecto: Adecuación, Plausibilidad, Justificación y Éxito.

8.1 Dimensión de Plausibilidad

Característica P1

Existen expertos, están disponibles y son cooperativos

Valor: **Si**

Justificación del valor: Varios miembros del staff del SEI tienen una rica experiencia en desarrollo en los sectores civiles y militares.

Característica P2

El experto es capaz de estructurar sus métodos y procedimientos de trabajo

Valor: **Todo**

Justificación del valor: El apunte muestra claramente como los expertos pudieron estructurar claramente el proceso.

Característica P3

La tarea está bien estructurada y se entiende.

Valor: **Todo**

Justificación del valor: La aplicación TBQ es bastante estructurada, aunque no completamente cerrada a cambios según el contexto.

Característica P4

Existen suficientes casos de prueba y sus soluciones asociadas.

Valor: **10**

Justificación del valor: Existen numerosos desarrollos anteriores en los que se trató de identificar los riesgos.

Característica P5

La tarea sólo depende de los conocimientos y no del sentido común.

Valor: **10**

Justificación del valor: Alguna situación muy particular puede necesitar del sentido común.

Característica	Peso	Valor	Intervalo Difuso				Peso*Valor				Peso/Valor			
P1	10	Sí	10	10	10	10	100	100	100	100	1	1	1	1
P2	7	Todo	7,8	8,8	10	10	54,6	61,6	70	70	0,9	0,8	0,7	0,7
P3	8	Todo	7,8	8,8	10	10	62,4	70,4	80	80	1	0,9	0,8	0,8
P4	10	10	10	10	10	10	100	100	100	100	1	1	1	1
P5	9	10	10	10	10	10	90	90	90	90	0,9	0,9	0,9	0,9
	44						407	422	440	440	4,8	4,6	4,4	4,4
				Resultado:			**9,19**	**9,57**	**10**	**10**				

8.2 Dimensión de Justificación

Característica J1

Resuelve una tarea útil y necesaria.

Valor: **Mucho**

Justificación del valor: la identificación de riesgos es muy importante en el desarrollo ya que puede ayudar a mitigarlos o por lo menos, a estar preparados para afrontarlos.

Característica J2

Se espera una alta tasa de recuperación de la inversión.

Valor: ***8***

Justificación del valor: la identificación de riesgos puede ahorrarnos grandes problemas ya entramos en conocimiento de los mismos tempranamente.

Característica J3

Hay escasez de experiencia humana.

Valor: **Poco**

Justificación del valor: los expertos tienen amplio conocimiento en desarrollo e identificación de riesgos.

Característica J4

Hay necesidad de tomar decisiones en situaciones críticas o ambientes hostiles, penosos, y, o, poco gratificantes.

Valor: **Poco**

Justificación del valor: no existe esa necesidad.

Característica J5

Hay necesidad de distribuir los conocimientos.

Valor: **Mucho**

Justificación del valor: es importante que el conocimiento esté al alcance de todos los usuarios del sistema.

Característica J6

Los conocimientos pueden perderse de no realizarse el sistema.

Valor: **Mucho**

Justificación del valor: no existen actualmente procesos claros para determinar los riesgos de un proyecto de software.

Característica J7

No existen soluciones alternativas.

Valor: **Si**

Justificación del valor: no existen actualmente procesos claros para determinar los riesgos de un proyecto de software.

Característica	Peso	Valor	Intervalo Difuso				Peso*Valor				Aprox. Numérica
J1	8	Mucho	5,6	6,6	7,8	8,8	45	53	62	70	57,6
J2	7	8	8	8	8	8	56	56	56	56	56
J3	6	Poco	1,2	2,2	3,4	4,4	7,2	13	20	26	16,8
J4	10	Poco	1,2	2,2	3,4	4,4	12	22	34	44	28
J5	10	Mucho	5,6	6,6	7,8	8,8	56	66	78	88	72
J6	10	Mucho	5,6	6,6	7,8	8,8	56	66	78	88	72
J7	8	Sí	10	10	10	10	80	80	80	80	80
										Máx:	80
				Resultado:			**10**	**10**	**10**	**10**	

8.3 Dimensión de Adecuación

Característica A1

La transferencia de experiencia entre humanos es factible.

Valor: **Mucho**

Justificación del valor: existe bibliografía sobre los riesgos en desarrollo de software.

Característica A2

La tarea requiere “experiencia”.

Valor: **Regular**

Justificación del valor: Debido al cuestionario, no es necesario ser un experto sobre riesgos en desarrollo o siquiera tener conocimientos muy específicos sobre el proyecto que se evalúa.

Característica A3

Los efectos de la introducción del SE no pueden preverse.

Valor: **Regular**

Justificación del valor: pueden preverse ya que contamos con varios expertos en el tema.

Característica A4

La tarea requiere razonamiento simbólico.

Valor: **Nada**

Justificación del valor: No es necesario.

Característica A5

La tarea requiere el uso de "heurísticas" para acotar el espacio de búsqueda.

Valor: **Nada**

Justificación del valor: No es necesario.

Característica A6

La tarea es de carácter público y más táctica que estratégica.

Valor: **Sí**

Justificación del valor: La tarea tiene mucho carácter público.

Característica A7

Se espera que la tarea continúe sin cambios significativos durante un largo período de tiempo.

Valor: **Mucho**

Justificación del valor: Debido al uso del cuestionario, no se esperan cambios significativos en la tarea.

Característica A8

Se necesitan varios niveles de abstracción en la resolución de la tarea.

Valor: **Todo**

Justificación del valor: Dado que existen diferentes áreas en las que se pueden detectar los riesgos.

Característica A9

El problema es relativamente simple o puede descomponerse en subproblemas.

Valor: **Mucho**

Justificación del valor: Es posible dividirlo en subproblemas, que serían las clases de la taxonomía.

Característica A10

El experto no sigue un proceso determinista en la resolución del problema.

Valor: **Si**

Justificación del valor: cada proyecto es distinto y su resolución requerirá nuevos análisis.

Característica A11

La tarea acepta la técnica del prototipado gradual.

Valor: **Si**

Justificación del valor: se han detectado subproblemas para los cuales se pueden hacer prototipos.

Característica A12

El experto resuelve el problema a veces con información incompleta o incierta.

Valor: **Regular**

Justificación del valor: Puede suceder que no se disponga de toda la información necesaria al momento de realizar la tarea.

Característica A13

Es conveniente justificar las soluciones adoptadas.

Valor: **Todo**

Justificación del valor: Para tener un panorama más completo de la situación.

Característica A14

La tarea requiere investigación básica.

Valor: **No**

Justificación del valor: La tarea no requiere investigación básica.

Característica A15

El sistema funcionará en “tiempo real” con otros programas o dispositivos.

Valor: **Nada**

Justificación del valor: No necesariamente, dado que no es un sistema de simulación.

Característica	Peso	Valor	Intervalo Difuso				Peso*Valor				Peso/Valor			
A1	7	Mucho	5,6	6,6	7,8	8,8	39,2	46,2	54,6	61,6	1,25	1,060606061	0,897435897	0,795454545
A2	10	Regular	3,4	4,4	5,6	6,6	34	44	56	66	2,941	2,272727273	1,785714286	1,515151515
A3	2	Regular	3,4	4,4	5,6	6,6	6,8	8,8	11,2	13,2	0,588	0,454545455	0,357142857	0,303030303
A4	5	Nada	0	0	1,2	2,2	0,05	0,05	6	11	500	500	4,166666667	2,272727273
A5	7	Nada	0	0	1,2	2,2	0,07	0,07	8,4	15,4	700	700	5,833333333	3,181818182
A6	8	Sí	10	10	10	10	80	80	80	80	0,8	0,8	0,8	0,8
A7	8	Mucho	5,6	6,6	7,8	8,8	44,8	52,8	62,4	70,4	1,429	1,212121212	1,025641026	0,909090909
A8	8	Todo	7,8	8,8	10	10	62,4	70,4	80	80	1,026	0,909090909	0,8	0,8
A9	6	Mucho	5,6	6,6	7,8	8,8	33,6	39,6	46,8	52,8	1,071	0,909090909	0,769230769	0,681818182
A10	3	Sí	10	10	10	10	30	30	30	30	0,3	0,3	0,3	0,3
A11	8	Sí	10	10	10	10	80	80	80	80	0,8	0,8	0,8	0,8
A12	3	Regular	3,4	4,4	5,6	6,6	10,2	13,2	16,8	19,8	0,882	0,681818182	0,535714286	0,454545455
A13	3	Todo	7,8	8,8	10	10	23,4	26,4	30	30	0,385	0,340909091	0,3	0,3
A14	10	No	0	0	0	0	0,1	0,1	0,1	0,1	1000	1000	1000	1000
A15	6	Nada	0	0	1,2	2,2	0,06	0,06	7,2	13,2	600	600	5	2,727272727
	94						444,7	491,68	569,5	623,5	2811	2809,740909	1023,370879	1015,840909
					Resultado		2,382	2,632	3,075	3,36276				

8.4 Dimensión de Éxito

Característica E1

Existe una ubicación idónea para el SE.

Valor: **Todo**

Justificación del valor: En la empresa que quiera utilizar el método para identificar los riesgos.

Característica E2

Problemas similares se han resuelto mediante INCO.

Valor: **No**

Justificación del valor: No hay conocimiento de otros sistemas de este estilo resueltos con INCO en Argentina.

Característica E3

El problema es similar a otros en los que resultó imposible aplicar esta tecnología.

Valor: **No**

Justificación del valor: No hay referencias.

Característica E4

La continuidad del proyecto está influenciada por vaivenes políticos.

Valor: **Nada**

Justificación del valor: No es influenciado por la política.

Característica E5

La inserción del sistema se efectúa sin traumas, es decir, apenas se interfiere en la rutina cotidiana.

Valor: **Nada**

Justificación del valor: No interferirá en la rutina cotidiana.

Característica E6

Se dispone de experiencia en INCO.

Valor: **Mucho**

Justificación del valor: Se dispone de material bibliográfico y asesoramiento de expertos.

Característica E7

Se dispone de los recursos humanos, hardware y software necesarios para el desarrollo e implementación del sistema.

Valor: **Todo**

Justificación del valor: Existen los recursos necesarios o pueden conseguirse.

Característica E8

El experto resuelve el problema en la actualidad.

Valor: **Todo**

Justificación del valor: El experto resuelve el problema en la actualidad.

Característica E9

La solución del problema es prioritaria para la institución.

Valor: **Mucho**

Justificación del valor: No es una prioridad de la institución pero como ayuda al buen desarrollo de sistemas sin duda es importante.

Característica E10

Las soluciones son explicables.

Valor: **Todo**

Justificación del valor: Son explicables las soluciones.

Característica E11

Los objetivos del sistema son claros y evaluables.

Valor: **Mucho**

Justificación del valor: Los objetivos son claros (identificar riesgos en el desarrollo de software) y evaluables (los expertos lo hacen con frecuencia) .

Característica E12

Los conocimientos están repartidos entre un conjunto de individuos.

Valor: **Regular**

Justificación del valor: Participan varios individuos pertenecientes a puestos variados, tanto de diferentes sectores como jerarquía en la empresa.

Característica E13

Los directivos, usuarios, expertos e IC están de acuerdo en las funcionalidades del SE.

Valor: **Mucho**

Justificación del valor: Sí, están de acuerdo con las funcionalidades que ofrecerá el sistema.

Característica E14

La actitud de los expertos ante el desarrollo del sistema es positiva y no se sienten amenazados por el proyecto.

Valor: **Mucho**

Justificación del valor: La mayoría piensa que el sistema los ayudará en gran medida en sus tareas, y no se ven amenazados ya que son necesarios para otras tareas.

Característica E15

Los expertos convergen en sus soluciones y métodos.

Valor: **Mucho**

Justificación del valor: Generalmente llegan a las mismas conclusiones.

Característica E16

Se acepta la planificación del proyecto propuesta por el IC.

Valor: **Si**

Justificación del valor: Los directivos y usuarios confían en la planificación realizada por el IC.

Característica E17

Existen limitaciones estrictas de tiempo en la realización del sistema.

Valor: **Regular**

Justificación del valor: Si bien no es estricto el tiempo en que se espera se finalice el proyecto. Se desea tenerlo funcionando en un tiempo razonable.

Característica E18

La dirección y usuarios apoyan los objetivos y directrices del proyecto.

Valor: **Mucho**

Justificación del valor: Ellos las apoyan en gran medida.

Característica E19

El nivel de formación requerido por los usuarios del sistema es elevado.

Valor: **Poco**

Justificación del valor: No es necesario que el nivel de formación sea elevado.

Característica E20

Las relaciones IC-Experto son fluidas.

Valor: **Mucho**

Justificación del valor: El experto accede de buena gana a las preguntas del IC.

Característica E21

El proyecto forma parte de un camino crítico con otros sistemas.

Valor: **No**

Justificación del valor: El proyecto no se utilizará en conjunto con otros proyectos.

Característica E22

Se efectuará una adecuada transferencia tecnológica.

Valor: **Todo**

Justificación del valor: La mayoría de los usuarios no tendrán inconvenientes en la utilización del nuevo sistema, ya que están acostumbrados al uso de nuevas tecnologías.

Característica E23

Lo que cuenta en la solución es la calidad de la respuesta.

Valor: **Si**

Justificación del valor: La calidad de la respuesta es fundamental para la decisión del ciclo de vida adecuado.

Característica	Peso	Valor	Intervalo Difuso				Peso*Valor				Peso/Valor			
E1	7	Todo	7,8	8,8	10	10	54,6	61,6	70	70	0,8974	0,7955	0,7	0,7
E2	8	No	0	0	0	0	0,08	0,08	0,08	0,08	800	800	800	800
E3	5	No	0	0	0	0	0,05	0,05	0,05	0,05	500	500	500	500
E4	9	Nada	0	1,2	2,2	3,4	0,09	10,8	19,8	30,6	900	7,5	4,0909	2,6471
E5	8	Nada	0	1,2	2,2	3,4	0,08	9,6	17,6	27,2	800	6,6667	3,6364	2,3529
E6	7	Mucho	5,6	6,6	7,8	8,8	39,2	46,2	54,6	61,6	1,25	1,0606	0,8974	0,7955
E7	4	Todo	7,8	8,8	10	10	31,2	35,2	40	40	0,5128	0,4545	0,4	0,4
E8	4	Todo	7,8	8,8	10	10	31,2	35,2	40	40	0,5128	0,4545	0,4	0,4
E9	8	Mucho	5,6	6,6	7,8	8,8	44,8	52,8	62,4	70,4	1,4286	1,2121	1,0256	0,9091
E10	5	Todo	7,8	8,8	10	10	39	44	50	50	0,641	0,5682	0,5	0,5
E11	6	Mucho	5,6	6,6	7,8	8,8	33,6	39,6	46,8	52,8	1,0714	0,9091	0,7692	0,6818
E12	7	Regular	3,4	4,4	5,6	6,6	23,8	30,8	39,2	46,2	2,0588	1,5909	1,25	1,0606
E13	4	Mucho	5,6	6,6	7,8	8,8	22,4	26,4	31,2	35,2	0,7143	0,6061	0,5128	0,4545
E14	8	Mucho	5,6	6,6	7,8	8,8	44,8	52,8	62,4	70,4	1,4286	1,2121	1,0256	0,9091
E15	5	Mucho	5,6	6,6	7,8	8,8	28	33	39	44	0,8929	0,7576	0,641	0,5682
E16	8	Sí	10	10	10	10	80	80	80	80	0,8	0,8	0,8	0,8
E17	6	Regular	3,4	4,4	5,6	6,6	20,4	26,4	33,6	39,6	1,7647	1,3636	1,0714	0,9091
E18	7	Mucho	5,6	6,6	7,8	8,8	39,2	46,2	54,6	61,6	1,25	1,0606	0,8974	0,7955
E19	2	Poco	1,2	2,2	3,4	4,4	2,4	4,4	6,8	8,8	1,6667	0,9091	0,5882	0,4545
E20	4	Mucho	5,6	6,6	7,8	8,8	22,4	26,4	31,2	35,2	0,7143	0,6061	0,5128	0,4545
E21	6	No	0	0	0	0	0,06	0,06	0,06	0,06	600	600	600	600
E22	8	Todo	7,8	8,8	10	10	62,4	70,4	80	80	1,0256	0,9091	0,8	0,8
E23	5	Sí	10	10	10	10	50	50	50	50	0,5	0,5	0,5	0,5
	141						669,8	782	909,4	993,8	3619,1	1929,9	1921	1917,1
				Resultado:			**2,39**	**2,81**	**3,26**	**3,56**				

8.5 Resultados

Dimensión	Peso	Valores Intervalo				Peso*Valor			
Plausibilidad	8	9,19	9,57	10	10	73,52	76,56	80	80
Justificación	3	10	10	10	10	30	30	30	30
Adecuación	8	2,38	2,63	3,08	3,36	19,04	21,04	24,64	26,88
Éxito	5	2,39	2,81	3,26	3,56	11,95	14,05	16,3	17,8
	24					134,5	141,7	150,9	154,7
Intervalo Resultado Final:						**5,6**	**5,9**	**6,29**	**6,45**

El resultado final del Análisis de Viabilidad es el promedio de los valores obtenidos de las cuatro dimensiones analizadas, el mismo es de 6,1. El proyecto es viable ya que supera el límite de 6 puntos que se utiliza en este método.

9. Análisis Estructural de textos

El análisis de texto consiste en la búsqueda, a través de la documentación, de determinados términos. El objetivo es extraer conceptos fundamentales del dominio para empezar a conocer el vocabulario de este. También se provee el significado de los términos, cuyo origen puede ser el mismo informe u otros textos especializados.

En el análisis estructural de texto se presenta en una tabla (ver **Tabla N° 1**) donde figuran términos propios del dominio del sistema (que el usuario puede no llegar a comprender) y su significado. El objetivo primordial de este análisis es proveerle al usuario una herramienta con el significado de muchos términos técnicos y específicos propios del dominio del sistema.

Los términos de la Tabla N° 1 que se encuentran en **negrita** son aquellos que engloban otros conceptos. De alguna manera podríamos hablar de conceptos complejos que encapsulan otros. Por ejemplo: el término **class (taxonomies)** hace referencia a los 3 tipos de taxonomías de riesgo que existen en el desarrollo de software que son: *Product Engineering*, *Development Environment* y *Program Constraints*.

9.1 Tabla N° 1

Término	Significado
Acceptance criteria	The criteria that a system or component must satisfy to be accepted by a user, customer, or other authorized entity
Acceptance testing	Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.
Analyze	Analysis is the conversion of risk data into risk decision-making information. Analysis provides the basis for the project manager to work on the “right” risks.
Application domain	Refers to the nature of the application. Two examples are real-time flight control systems and management information systems.
Attributte	• Stability. • Scale.

	• Formality. • Product Control. • Schedule. • Facilities.
Audit	An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria.
Availability	The relative time that an operational product must be available for use. Usually expressed as the ratio of time available for use to some total time period or as specific hours of operation.
Baseline	A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
Baseline management	In configuration management, the application of technical and administrative direction to designate the documents and changes to those documents that formally identify and establish baselines at specific times during the life cycle of a configuration item.
Benchmark	A standard against which measurements or comparisons can be made.
Change control	A part of configuration management that reviews, approves, and tracks progress of alterations in the configuration of a configuration item delivered, to be delivered, or under formal development, after formal establishment of its configuration identification.
Class (Taxonomies)	• Product Engineering. • Development Environment. • Program Constraints.
Command	A signal that initiates an operation defined by an instruction.
Communicate	Risk communication lies at the center of the model to emphasize both its pervasiveness and its criticality.

	Without effective communication, no risk management approach can be viable. While communication facilitates interaction among the elements of the model, there are higher level communications to consider as well. To be analyzed and managed correctly, risks must be communicated to and between the appropriate organizational levels and entities. This includes levels within the development project and organization, within the customer organization, and most especially, across that threshold between the developer, the customer, and, where different, the user. Because communication is pervasive, our approach is to address it as integral to every risk management activity and not as something performed outside of, and as a supplement to, other activities.
Communication	The process of transmitting and receiving ideas, information, and messages.
Configuration	In configuration management, the functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product.
Configuration management	A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a controlled item, control changes to a configuration item and its documentation, and record and report change processing and implementation status.
Configuration management function	The organizational element charged with configuration management.
Configuration management system	The processes, procedures, and tools used by the development organization to accomplish configuration management.
Control	To hold in restraint; check. To verify or regulate by systematic comparison.
Cost	A loss, sacrifice, or penalty.

COTS (commercial off-the-shelf)	A type of non-developmental software that is supplied by commercial sources.
Critical design review (CDR)	A review conducted to verify that the detailed design of one or more configuration items satisfy specified requirements; to establish the compatibility among the configuration items and other items of equipment, facilities, software, and personnel; to assess risk areas for each configuration item; and, as applicable, to assess the results of producability analyses, review preliminary hardware product specifications, evaluate preliminary test planning, and evaluate the adequacy of preliminary operation and support documents.
Customer	The person or organization receiving a product or service. There may be many different customers for individual organizations within a program structure. Government program offices may view the customer as the user organization for which they are managing the project. Contractors may view the program office as well as the user organization as customers.
Data	Factual information, especially information organized for analysis or used to make decisions. Numerical information suitable for processing by computer.
Database	A collection of data arranged for ease of search and retrieval.
Design specifications	A document that prescribes the form, parts, and details of the product according to a plan.
Design-to-cost	Bidding a selected, reduced set of requirements to meet cost objectives.
Detailed design	The process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to be implemented.
Developer	Person who makes something available and usable.
Development computer	The hardware and supporting software system used for

	software development.
Development Environment (class)	The development environment class is concerned with the project environment in which a software product is engineered. This environment consists of the following elements: • **Development Process**. The definition, planning, documentation, suitability, enforcement, and communication of the methods and procedures used to develop the product. • **Development System**. The tools and supporting equipment used in product development, such as computer-aided software engineering (CASE) tools, simulators, compilers, and host computer systems. • **Management Process**. The planning, monitoring, and controlling of budgets and schedules; controlling factors involved in defining, implementing, and testing the product; the project manager's experience in software development, management, and the product domain; and the manager's expertise in dealing with external organizations including customers, senior management, matrix management, and other contractors. • **Management Methods**. The methods, tools, and supporting equipment that will be used to manage and control the product development, such as monitoring tools, personnel management, quality assurance, and configuration management. • **Work Environment**. The general environment within which the work will be performed, including the attitudes of people and the levels of cooperation, communication, and morale.
Development facilities	The office space, furnishings, and equipment that support the development staff.
Development model	The abstract visualization of how the software development functions (such as requirements definition,

	design, code, test, and implementation) are organized. Typical models are the waterfall model, the iterative model, and the spiral model.
Development process	The implemented process for managing the development of the deliverable product. For software, the development process includes the following major activities: translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use. These activities may overlap and may be applied iteratively or recursively.
Development sites	The locations at which development work is being conducted.
Development system	The hardware and software tools and supporting equipment that will be used in product development including such items as computer-aided software engineering (CASE) tools, compilers, configuration management systems, and the like.
Element	• Requirements. • Engineering Specialities. • Development Process. • Work Environment. • Resources. • Program Interfaces.
External dependencies	Any deliverables from other organizations that are critical to a product's success.
External interfaces	The points where the software system under development interacts with other systems, sites, or people.
Framework	Is a basic conceptual structure used to solve or address complex issues. Is used in research to outline possible courses of action or to present a preferred approach to an idea or thought.
Hardware specifications	A document that prescribes the functions, materials,

	dimensions, and quality that a hardware item must meet.
Identification Conclusion	In concluding the risk identification, it was necessary to provide feedback to all participants. This was done through a results briefing. This briefing consists of all identified issues, and some suggestions on the next steps in managing the identified issues. The intent of the briefing is to provide the participants with feedback on the results of their efforts.
Identify	Before risks can be managed, they must be identified. Identification surfaces risks before they become problems and adversely affect a project. The SEI has developed techniques for surfacing risks by the application of a disciplined and systematic process that encourages project personnel to raise concerns and issues for subsequent analysis. One such technique, the taxonomy-based questionnaire, is described in subsequent chapters of this report.
IEEE	Institute of Electrical and Electronics Engineers, Inc.
Implementation	The act of preparing a product for use by the customer.
Information System	An organized collection, storage, and presentation system of data and other knowledge for decision making, progress reporting, and for planning and evaluation of programs. It can be either manual or computerized, or a combination of both.
Integration	The act of assembling individual hardware and/or software components into a usable whole.
Integration environment	The hardware, software, and supporting tools that will be used to support product integration.
Integration testing	Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them.
Internal interfaces	The points where the software system under development interacts with other components of the system under

	development.
Issues	For field test purposes, an abbreviated ranking exercise was conducted with a management team on a selected subset of identified issues. The results briefing included a description of the method of selecting risks for ranking and the results of that ranking.
Long-term issues	Issues of strategic importance to the project that can be compromised in the heat of battle. Issues such as employee training and development, establishing and improving processes and procedures, and similar activities are important to the long term viability of the project and the organization.
Mainteinance	To keep in good repair. To provide for; support.
Management	The person or persons who manage an organization. Executive ability.
Non-developmental software (NDS)	Deliverable software that is not developed under the contract but is provided by the contractor, the Government, or a third party. NDS may be referred to as reusable software, Government furnished software, or commercially available software, depending on its source.
Orange Book	A security standard set by the U.S. Government as described in Federal Criteria for Information Technology Security, Volume 1, December 1992.
Plan	Planning turns risk information into decisions and actions (both present and future). Planning involves developing actions to address individual risks, prioritizing risk actions, and creating an integrated risk management plan.The plan for a specific risk could take many forms. For example: • Mitigate the impact of the risk by developing a contingency plan (along with an identified triggering event) should the risk occur.

	• Avoid a risk by changing the product design or the development process. • Accept the risk and take no further action, thus accepting the consequences if the risk occurs. • Study the risk further to acquire more information and better determine the characteristics of the risk to enable decision making. The key to risk action planning is to consider the future consequences of a decision made today.
Preliminary design	The process of analyzing design alternatives and defining the architecture, components, interfaces, and timing and sizing estimates for a system or component.
Procedure	A written description of a course of action to be taken to perform a given task.
Process	A sequence of steps performed for a given purpose; for example, the software development process.
Product Enginnering (class)	The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The class focuses on the work to be performed, and includes the following elements: • **Requirements**. The definition of what the software product is to do, the needs it must meet, how it is to behave, and how it will be used. This element also addresses the feasibility of developing the product and the scale of the effort. • **Design**. The translation of requirements into an effective design within project and operational constraints. • **Code and Unit Test.** The translation of software designs into code that satisfies the requirements allocated to individual units. • **Integration and Test**. The integration of units into a

	working system and the validation that the software product performs as required. • **Engineering Specialities**. Product requirements or development activities that may need specialized expertise such as safety, security, and reliability.
Product integration	The act of assembling individual hardware and software components into a functional whole.
Project	A plan or proposal; scheme. An undertaking requiring concerted effort.
Program Constraints (class)	The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but can still have major effects on its success. Program constraints include the following elements: • **Resources**. The external constraints imposed on schedule, staff, budget, or facilities. • **Contract**. The terms and conditions of the project contract. • **Program Interfaces**. The external interfaces to customers, other contractors, corporate management, and vendors.
Quality	A trait or characteristic; property. Essential character; nature. Degree or grade of excellence.
Re-engineering	The practice of adapting existing software artifacts or systems to perform new or enhanced functions. Re-engineered artifacts may be substantially different from the existing artifact.
Reliability	The degree of dependability that an operational product must meet. Usually expressed as the average time to failure.
Requirements analysis	(1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements. (2) The process of studying and refining system,

	hardware, or software requirements.
Reusing	Hardware or software developed in response to the requirements of one application that can be used, in whole or in part, to satisfy the requirements of another application.
Risk	The possibility of suffering harm or loss; danger. A factor, element, or course involving uncertain danger. To expose to a chance of loss or damage.
Risk Identification	The risk identification session began with a briefing to all risk identification participants consisting of a description of the TBQ method and a summary of the schedule and process to be followed over the duration of the risk identification. In the interests of minimizing the overall time spent on identification, all participants selected by the project were asked to attend. Attendees may also include personnel who might take an active role in future risk identification or other risk management activities for the project.
Safety	The degree to which the software product minimizes the potential for hazardous conditions during its operational mission.
Schedule	A timetable. A production plan. A list of items. A program of events or appointments.
Security	The degree to which a software product is safe from unauthorized use.
SEI risk management paradigm	Shows the different activities composing software development risk management. The paradigm is represented as a circle to emphasize that risk management is a continuous process while the arrows show the logical and temporal flow of information between the activities in risk management. Communication is placed in the center of the paradigm because it is both the conduit

	through which all information flows and, often, is the major obstacle to risk management. In essence, the paradigm is a framework for software risk management. From this framework, a project may structure a risk management practice best fitting into its project management structure.
Software	Computer programs; instructions that cause the hardware—the machines—to do work. Software as a whole can be divided into a number of categories based on the types of work done by programs.
Software Development Risk Taxonomy	Central to the risk identification method is the software development taxonomy. The taxonomy provides a framework for organizing and studying the breadth of software development issues. Hence, it serves as the basis for eliciting and organizing the full breadth of software development risks—both technical and non-technical. The taxonomy also provides a consistent framework for the development of other risk management methods and activities. The software taxonomy is organized into three major classes. 1. **Product Engineering**. The technical aspects of the work to be accomplished. 2. **Development Environment**. The methods, procedures, and tools used to produce the product. 3. **Program Constraints**. The contractual, organizational, and operational factors within which the software is developed but which are generally outside of the direct control of the local management. These taxonomic classes are further divided into elements and each element is characterized by its attributes.
Software architecture	The organizational structure of the software or module.
Software life cycle	The period of time that begins when a software product is

	conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase.
Software requirement	A condition or capability that must be met by software needed by a user to solve a problem or achieve an objective.
Software requirements specification (SRS)	Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.
System integration	The act of assembling hardware and/or software components into a deliverable product.
System requirement	A condition or capability that must be met or possessed by a system or system component to satisfy a condition or capability needed by a user to solve a problem.
System testing	Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.
Target computer	The hardware and supporting software system that will actually be used when the software system is fielded.
Taxonomy	A taxonomy is a scheme that partitions a body of knowledge and defines the relationships among the pieces. It is used for classifying and understanding the body of knowledge.
TBDs	Requirements in formal requirements statements that are to be defined.
Test specifications	A document that prescribes the process and procedures to be used to verify that a product meets its requirements.
TQB (taxonomy-based questionnaire)	The TBQ consists of questions under each taxonomic attribute designed to elicit the range of risks and concerns potentially affecting the software product.

Traceability	The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another.
Traceability mechanism	Processes and procedures (manual and/or automated) that map all software components and artifacts from source requirements through test cases.
Track	Tracking consists of monitoring the status of risks and actions taken to ameliorate risks. Appropriate risk metrics are identified and monitored to enable the evaluation of the status of risks themselves and of risk mitigation plans. Tracking serves as the “watch dog” function of management.
Transition plan	A plan (documented in the Computer Resources Integrated Support Document) specifying how products are to be transitioned from development to support.
Track	Tracking consists of monitoring the status of risks and actions taken to ameliorate risks. Appropriate risk metrics are identified and monitored to enable the evaluation of the status of risks themselves and of risk mitigation plans. Tracking serves as the “watch dog” function of management.
Transition plan	A plan (documented in the Computer Resources Integrated Support Document) specifying how products are to be transitioned from development to support.
unit	(1) A separately testable element specified in the design of a computer software component. (2) A logically separable part of a computer program. (3) A software component that is not subdivided into other components.
unit testing	Testing of individual hardware or software units or groups of related units.

10. Cálculo del Emparrillado

El Emparrillado está basado en la "Teoría de la Construcción Personal". Esta técnica de educción es un modelo del pensamiento humano, que se basa en el concepto de que cada persona tiene su propia visión del mundo que lo rodea.

Básicamente, es un test de clasificación en el cual se vincula una lista de elementos sobre la base de un conjunto bipolar de características.

En este SE los elementos que se comparan son las clases de la Taxonomía creada por el SEI.

Para el cálculo del emparrillado se consideraron los siguientes elementos, que se corresponden con las *clases* de la Taxonomía del SEI:

1. *E1*: Product Engineering
2. *E2*: Development Environment
3. *E3*: Program Constraints

Y las siguientes características:

1. *C1*: Intelectual/Física: si la clase está relacionada con algo teórico o algo más práctico.
2. *C2*: Interna/Externa: si la clase está relacionada con tareas de control directo de la empresa, o fuera del control directo.
3. *C3*: Uso IDEs en Resolución (Poco/Mucho): si la clase permite el uso de software con interfaz gráfica para llevarla a cabo.
4. *C4*: Técnica/No Técnica: si la clase está relacionada con tareas pertenecientes a la ingeniería del software, o no.
5. *C5*: Desarrollo/Management: si la clase está relacionada con tareas de programación o de planificación.

Parrilla Evaluada

	E1	E2	E3
C1	2	2	4
C2	1	2	4
C3	5	3	2
C4	5	4	2
C5	5	3	1

10.1 Clasificación de los Elementos

Matriz distancia entre elementos

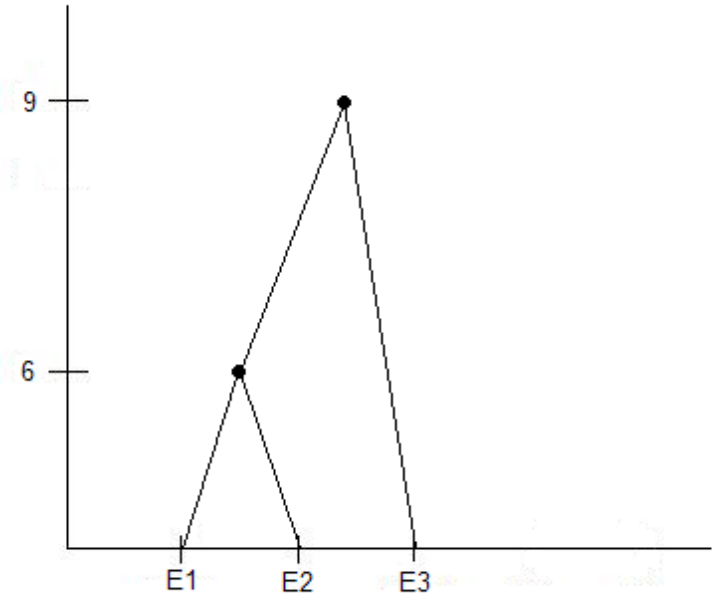
	E1	E2	E3
E1		6	15
E2			9
E3			

Dado que la distancia mínima se encuentra entre los elementos E1 y E2, los mismos se agrupan:

	Distancia con E1	Distancia con E2	Menor Distancia
E3	15	9	9

	E1 - E2	E3
E1 - E2		9
E3		

10.1.1 Árbol de Elementos



10.1.2 Análisis de los resultados

Un análisis del árbol nos permite ver que los elementos más parecidos son Product Engineering y Development Environment, aunque esto de todas maneras es relativo ya que están separados por una distancia de 6.

La distancia entre los dos primeros y el último, Program Constraints, puede ser explicada por su definición de ser las cosas externas o que están fuera del control directo del proyecto, a diferencia de las anteriores que si están bajo el control de personas de la empresa que quiere identificar riesgos.

10.2 Clasificación de Características

Matriz Triangular Superior

	C1	C2	C3	C4	C5
C1		1	6	7	7
C2			7	7	8
C3				1	1
C4					2
C5					

Matriz de Opuestos

	E1	E2	E3
No C1	4	4	2
No C2	5	4	2
No C3	1	3	4
No C4	1	2	4
No C5	1	3	5

Matriz distancia entre Características

	C1	C2	C3	C4	C5
C1		1	6	7	7
C2	7		7	7	8
C3	2	1		1	1
C4	1	0	7		2
C5	3	2	7	8	

A partir de los valores obtenidos en ésta matriz se arma la matriz de distancias mínimas con los mínimos valores para cada par de características.

Comb.	Dist.	Comb.	Dist.	Menor
c1-c2	1	c1-no c2	7	1
c1-c3	6	c1-no c3	2	2
c1-c4	7	c1-no c4	1	1
c1-c5	7	c1-no c5	3	3
c2-c3	7	c2-no c3	1	1
c2-c4	8	c2-no c4	0	0
c2-c5	8	c2-no c5	2	2
c3-c4	1	c3-no c4	7	1
c3-c5	1	c3-no c5	7	1
c4-c5	2	c4-no c5	8	2

Matriz distancias Mínimas

	No C1	No C2	C3	C4	C5
No C1		1	2	1	3
No C2			2	0	2
C3				1	1
C4					2
C5					

Dado que la mínima distancia se encuentra entre las características 'No C2' y C4 se agrupan las mismas. Primero se muestra la tabla que permite obtener el valor que cada elemento restante tendrá con el resulta de la combinación de los dos elementos anteriores:

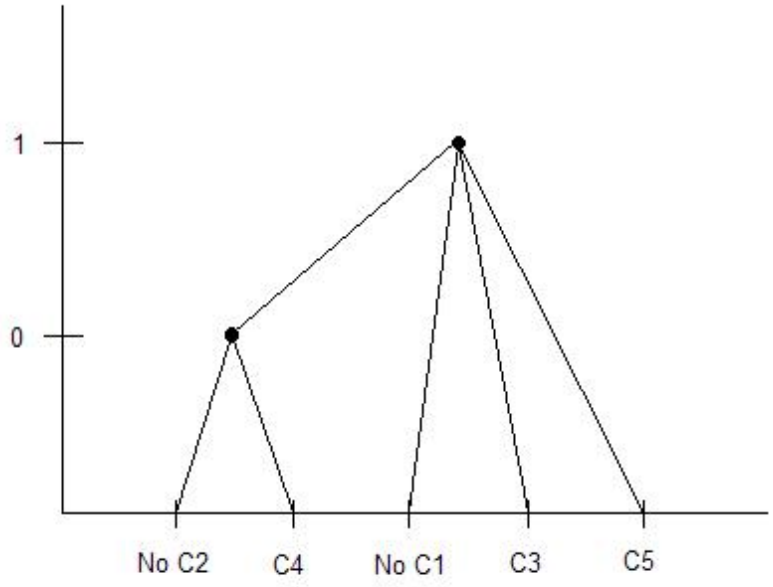
	D No C2	D C4	Menor
No C1	1	1	1
C3	2	1	1
C5	2	2	2

La nueva tabla es la siguiente:

	No C1	No C2 - C4	C3	C5
No C1		1	2	3
No C2 - C4			1	2
C3				1
C5				

En esta iteración se unen los elementos 'No C2 – C4' con 'No C1', C3 y C5. Como no quedan más características, esta termina siendo la tabla final de características.

10.2.1 Árbol de Características



10.2.2 Análisis de los resultados

En base al árbol podemos decir que C2 (Interna / Externa) y C4 (Técnica / No Técnica) son características opuestas ya que la negación de la primera tiene distancia cero con la segunda.
Algo parecido podría decirse entre C1 (Intelectual / Física) y C3 (Uso de IDEs en resolución (Poco / Mucho)) y C5 (Desarrollo / Management) ya que las tres se encuentran a distancia de las anteriores.
Obviando los opuestos mencionados, se puede decir que las características son bastante similares entre sí, ya que las distancias son bastante pequeñas, al punto que ninguna llega a dos.

11. Conocimientos Fácticos

En esta parte se especifican los datos que el sistema necesita de entrada, salida y para realizar las distintas tareas. Se identifican conceptos, atributos y valores asociados. Para ellos, se crean un glosario de términos, la tabla de **concepto – atributo – valor** (**TCAV**) y el mapa de relaciones.

El conocimiento fáctico engloba todos los atributos, valores, sujetos, verbos, elementos, sustantivos, conceptos, etc. que forman parte del dominio en el cual estamos trabajando. En este caso en particular el dominio es la identificación de las distintas taxonomías de riesgo que se pueden dar en el desarrollo de software.

11.1 Glosario de Términos

En el glosario de términos se provee una tabla de términos y sus respectivas definiciones (ver **Tabla N° 2**). Estos términos son aquellos que el usuario debe saber para poder comprender el dominio del sistema. Por ser estos términos parte del dominio del sistema definen muchos de los atributos de cada una de las taxonomías de riesgo para el desarrollo de software. Es una herramienta que ayuda a poder comprender con mayor profundidad determinados conceptos que pueden prestar a confusión.

Los términos que poseen llamadas son conceptos que se encuentran repetidos pero cuyos contextos son diferentes. Al final de la Tabla N° 2 se encuentran explicados con mayor exactitud en un anexo (**Anexo de la Tabla N° 2**)

11.1.1 Tabla N° 2

Término	Definición
Associate Contractors	The presence of associate contractors may introduce risks due to conflicting political agendas, problems of interfaces to systems being developed by outside organizations, or lack of cooperation in coordinating schedules and configuration changes.
Budget	This attribute refers to the stability of the budget with respect to internal and external events or dependencies and the viability of estimates and planning for all phases and aspects of the program.
Capacity	Risks associated with the capacity of the development system may result from too few workstations, insufficient processing power or database storage, or other inadequacies in equipment to support parallel activities for development, test, and support activities.
Clarity	This attribute refers to ambiguously or imprecisely written individual requirements that are not resolved until late in the development phase. This lack of a mutual contractor and customer understanding may require re-work to meet the customer intent for a requirement.
Code and Unit Test	Attributes of this element are associated with the quality

	and stability of software or interface specifications, and constraints that may present implementation or test difficulties.
Coding/Implementation	This attribute addresses the implications of implementation constraints. Some of these are: target hardware that is marginal or inadequate with regard to speed, architecture, memory size or external storage capacity; required implementation languages or methods; or differences between the development and target hardware.
Communication	Risks that result from poor communication are due to lack of knowledge of the system mission, requirements, and design goals and methods, or to lack of information about the importance of program goals to the company or the project.
Completeness	Missing or incompletely specified requirements may appear in many forms, such as a requirements document with many functions or parameters “to be defined”; requirements that are not specified adequately to develop acceptance criteria, or inadvertently omitted requirements. When missing information is not supplied in a timely manner, implementation may be based on contractor assumptions that differ from customer expectations. When customer expectations are not documented in the specification, they are not budgeted into the cost and schedule.
Configuration Management	The configuration management (CM) attribute addresses both staffing and tools for the CM function as well as the complexity of the required CM process with respect to such factors as multiple development and installation sites and product coordination with existing, possibly changing, systems.
Contract	Risks associated with the program contract are classified according to contract type, restrictions, and dependencies.

Cooperation	The period of time that begins when a software The cooperation attribute addresses lack of team spirit among development staff both within and across work groups and the failure of all management levels to demonstrate that best efforts are being made to remove barriers to efficient accomplishment of work.
Corporate Management	Risks in the corporate management area include poor communication and direction from senior management as well as non-optimum levels of support.
Customer	The customer attribute refers to the customer's level of skill and experience in the technical or application domain of the program as well as difficult working relationships or poor mechanisms for attaining customer agreement and approvals, not having access to certain customer factions, or not being able to communicate with the customer in a forthright manner.
Deliverability	Some contracts require delivery of the development system. Risks may result from neglecting to bid and allocate resources to ensure that the development system meets all deliverable requirements.
Dependencies	This attribute refers to the possible contractual dependencies on outside contractors or vendors, customer-furnished equipment or software, or other outside products and services.
Design	The attributes of the design element cover the design and feasibility of algorithms, functions or performance requirements, and internal and external product interfaces. Difficulty in testing may begin here with failure to work to testable requirements or to include test features in the design. The following attributes characterize the design element.
Development Environment	The development environment class addresses the project environment and the process used to engineer a software product. This environment includes the development

	process and system, management methods, and work environment. These environmental elements are characterized below by their component attributes.
Development Process	The development process element refers to the process by which the contractor proposes to satisfy the customer's requirements. The process is the sequence of steps—the inputs, outputs, actions, validation criteria, and monitoring activities—leading from the initial requirement specification to the final delivered product. The development process includes such phases as requirements analysis, product definition, product creation, testing, and delivery. It includes both general management processes such as costing, schedule tracking, and personnel assignment, and also project-specific processes such as feasibility studies, design reviews, and regression testing. This element groups risks that result from a development process that is inadequately planned, defined and documented; that is not suited to the activities necessary to accomplish the project goals; and that is poorly communicated to the staff and lacks enforced usage.
Development System	The development system element addresses the hardware and software tools and supporting equipment used in product development. This includes computer aided software engineering tools, simulators, compilers, test equipment, and host computer systems.
Difficulty	The difficulty attribute refers to functional or design requirements that may be extremely difficult to realize. Systems engineering may design a system architecture difficult to implement, or requirements analysis may have been based on optimistic design assumptions. The difficulty attribute differs from design feasibility in that it does not proceed from pre-ordained algorithms or designs.
Engineering Specialities	The engineering specialty requirements are treated

	separately from the general requirements element primarily because they are often addressed by specialists who may not be full time on the program. This taxonomic separation is a device to ensure that these specialists are called in to analyze the risks associated with their areas of expertise.
Environment	The integration and test environment includes the hardware and software support facilities and adequate test cases reflecting realistic operational scenarios and realistic test data and conditions. This attribute addresses the adequacy of this environment to enable integration in a realistic environment or to fully test all functional and performance requirements.
Facilities	This attribute refers to the adequacy of the program facilities for development, integration, and testing of the product.
Familiarity (1)	Familiarity with the development process covers knowledge of, experience in, and comfort with the prescribed process.
Familiarity (2)	Development system familiarity depends on prior use of the system by the company and by project personnel as well as adequate training for new users.
Feasibility (1)	The feasibility attribute refers to the difficulty of implementing a single technical or operational requirement, or of simultaneously meeting conflicting requirements. Sometimes two requirements by themselves are feasible, but together are not; they cannot both exist in the same product at the same time. Also included is the ability to determine an adequate qualification method for demonstration that the system satisfies the requirement.
Feasibility (2)	The feasibility attribute of the code and unit test element addresses possible difficulties that may arise from poor design or design specification or from inherently difficult implementation needs.

	For example, the design may not have quality attributes such as module cohesiveness or interface minimization; the size of the modules may contribute complexity; the design may not be specified in sufficient detail, requiring the programmer to make assumptions or design decisions during coding; or the design and interface specifications may be changing, perhaps without an approved detailed design baseline; and the use of developmental hardware may make an additional contribution to inadequate or unstable interface specification. Or, the nature of the system itself may aggravate the difficulty and complexity of the coding task.
Formality	Formality of the development process is a function of the degree to which a consistent process is defined, documented, and communicated for all aspects and phases of the development.
Functionality	This attribute covers functional requirements that may not submit to a feasible design, or use of specified algorithms or designs without a high degree of certainty that they will satisfy their source requirements. Algorithm and design studies may not have used appropriate investigation techniques or may show marginal feasibility.
Hardware Constraints	This attribute covers target hardware with respect to system and processor architecture, and the dependence on hardware to meet system and software performance requirements. These constraints may include throughput or memory speeds, real-time response capability, database access or capacity limitations, insufficient reliability, unsuitability to system function, or insufficiency in the amount of specified hardware.
Human Factors	Meeting human factors requirements is dependent on understanding the operational environment of the installed system and agreement with various customer and user factions on a mutual understanding of the expectations

	embodied in the human factors requirements. It is difficult to convey this understanding in a written specification. Mutual agreement on the human interface may require continuous prototyping and demonstration to various customer factions.
Integration and Test	This element covers integration and test planning, execution, and facilities for both the contractual product and for the integration of the product into the system or site environment.
Interfaces	This attribute covers all hardware and software interfaces that are within the scope of the development program, including interfaces between configuration items, and the techniques for defining and managing the interfaces. Special note is taken of non-developmental software and developmental hardware interfaces.
Maintainability	Maintainability may be impaired by poor software architecture, design, code, or documentation resulting from undefined or un-enforced standards, or from neglecting to analyze the system from a maintenance point of view.
Management Experience	This attribute refers to the experience of all levels of managers with respect to management, software development management, the application domain, the scale and complexity of the system and program, the selected development process, and hands-on development of software.
Management Methods	This element refers to methods for managing both the development of the product and program personnel. These include quality assurance, configuration management, staff development with respect to program needs, and maintaining communication about program status and needs.
Management Process	The management process element pertains to risks associated with planning, monitoring, and controlling budget and schedule; with controlling factors involved in

	defining, implementing, and testing the product; with managing project personnel; and with handling external organizations including the customer, senior management, matrix management, and other contractors.
Monitoring	The monitoring includes the activities of obtaining and acting upon status reports, allocating status information to the appropriate program organizations, and maintaining and using progress metrics.
Morale	Risks that result from low morale range across low levels of enthusiasm and thus low performance, productivity or creativity; anger that may result in intentional damage to the project or the product; mass exodus of staff from the project; and a reputation within the company that makes it difficult to recruit.
Non-Developmental Software	Since non-developmental software (NDS) is not designed to system requirements, but selected as a "best fit," it may not conform precisely to performance, operability or supportability requirements. The customer may not accept vendor or developer test and reliability data to demonstrate satisfaction of the requirements allocated to NDS. It may then be difficult to produce this data to satisfy acceptance criteria and within the estimated NDS test budget. Requirements changes may necessitate re-engineering or reliance on vendors for special purpose upgrades.
Performance	The performance attribute refers to time-critical performance: user and real-time response requirements, throughput requirements, performance analyses, and performance modeling throughout the development cycle.
Personnel Management	Personnel management refers to selection and training of program members and ensuring that they: take part in planning and customer interaction for their areas of responsibility; work according to plan; and receive the help they need or ask for to carry out their responsibilities.
Planning	The planning attribute addresses risks associated with

	developing a well-defined plan that is responsive to contingencies as well as long-range goals and that was formulated with the input and acquiescence of those affected by it. Also addressed are managing according to the plan and formally modifying the plan when changes are necessary.
Politics	Political risks may accrue from relationships with the company, customer, associate contractors or subcontractors, and may affect technical decisions.
Precedent	The precedent attribute concerns capabilities that have not been successfully implemented in any existing systems or are beyond the experience of program personnel or of the company. The degree of risk depends on allocation of additional schedule and budget to determine the feasibility of their implementation; contingency plans in case the requirements are not feasible as stated; and flexibility in the contract to allocate implementation budget and schedule based on the outcome of the feasibility study. Even when unprecedented requirements are feasible, there may still be a risk of underestimating the difficulty of implementation and committing to an inadequate budget and schedule.
Prime Contractor	When the program is a subcontract, risks may arise from poorly defined task definitions, complex reporting arrangements, or dependencies on technical or programmatic information.
Process Control	Process control refers not only to ensuring usage of the defined process by program personnel, but also to the measurement and improvement of the process based on observation with respect to quality and productivity goals. Control may be complicated due to distributed development sites.
Product	The product integration attribute refers to integration of the

	software components to each other and to the target hardware, and testing of the contractually deliverable product. Factors that may affect this are internal interface specifications for either hardware or software, testability of requirements, negotiation of customer agreement on test criteria, adequacy of test specifications, and sufficiency of time for integration and test.
Product Control	Product control is dependent on traceability of requirements from the source specification through implementation such that the product test will demonstrate the source requirements. The change control process makes use of the traceability mechanism in impact analyses and reflects all resultant document modifications including interface and test documentation.
Product engineering	Product engineering refers to the system engineering and software engineering activities involved in creating a system that satisfies specified requirements and customer expectations. These activities include system and software requirements analysis and specification, software design and implementation, integration of hardware and software components, and software and system test. The elements of this class cover traditional software engineering activities. They comprise those technical factors associated with the deliverable product itself, independent of the processes or tools used to produce it or the constraints imposed by finite resources or external factors beyond program control. Product engineering risks generally result from requirements that are technically difficult or impossible to implement, often in combination with inability to negotiate relaxed requirements or revised budgets and schedules;

	from inadequate analysis of requirements or design specification; or from poor quality design or coding specifications.
Program Constraints	Program constraints refer to the “externals” of the project. These are factors that may be outside the control of the project but can still have major effects on its success or constitute sources of substantial risk.
Program Interfaces (1)	This attribute refers to the interactions of managers at all levels with program personnel at all levels, and with external personnel such as the customer, senior management, and peer managers.
Program Interfaces (2)	This element consists of the various interfaces with entities and organizations outside the development program itself.
Project Organization	This attribute addresses the effectiveness of the program organization, the effective definition of roles and responsibilities, and the assurance that these roles and lines of authority are understood by program personnel.
Quality Assurance	The quality assurance attribute refers to the procedures instituted for ensuring both that contractual processes and standards are implemented properly for all program activities, and that the quality assurance function is adequately staffed to perform its duties.
Quality Attitude	This attribute refers to the tendency of program personnel to do quality work in general and to conform to specific quality standards for the program and product.
Reliability (1)	System reliability or availability requirements may be affected by hardware not meeting its reliability specifications or system complexity that aggravates difficulties in meeting recovery timelines. Reliability or availability requirements allocated to software may be stated in absolute terms, rather than as separable from hardware and independently testable.
Reliability (2)	Development system reliability is a measure of whether the needed components of the development system are

	available and working properly whenever required by any program personnel.
Requirements	Attributes of the requirements element cover both the quality of the requirements specification and also the difficulty of implementing a system that satisfies the requirements.
Resources	This element addresses resources for which the program is dependent on factors outside program control to obtain and maintain. These include schedule, staff, budget, and facilities.
Restrictions	Contract restrictions and restraints refer to contractual directives to, for example, use specific development methods or equipment and the resultant complications such as acquisition of data rights for use of non-developmental software.
Safety	This attribute addresses the difficulty of implementing allocated safety requirements and also the potential difficulty of demonstrating satisfaction of requirements by faithful simulation of the unsafe conditions and corrective actions. Full demonstration may not be possible until the system is installed and operational.
Scale	This attribute covers both technical and management challenges presented by large complex systems development. Technical challenges include satisfaction of timing, scheduling and response requirements, communication among processors, complexity of system integration, analysis of inter-component dependencies, and impact due to changes in requirements. Management of a large number of tasks and people introduces a complexity in such areas as project organization, delegation of responsibilities, communication among management and peers, and configuration management.

Schedule	This attribute refers to the stability of the schedule with respect to internal and external events or dependencies and the viability of estimates and planning for all phases and aspects of the program.
Security	This attribute addresses lack of experience in implementing the required level of system security that may result in underestimation of the effort required for rigorous verification methods, certification and accreditation, and secure or trusted development process logistics; developing to unprecedented requirements; and dependencies on delivery of certified hardware or software.
Specifications	This attribute addresses specifications for the system, hardware, software, interface, or test requirements or design at any level with respect to feasibility of implementation and the quality attributes of stability, completeness, clarity, and verifiability.
Stability	The stability attribute refers to the degree to which the requirements are changing and the possible effect changing requirements and external interfaces will have on the quality, functionality, schedule, design, integration, and testing of the product being built. The attribute also includes issues that arise from the inability to control rapidly changing requirements. For example, impact analyses may be inaccurate because it is impossible to define the baseline against which the changes will be implemented.
Staff	This attribute refers to the stability and adequacy of the staff in terms of numbers and skill levels, their experience and skills in the required technical areas and application domain, and their availability when needed.
Subcontractors	The presence of subcontractors may introduce risks due to inadequate task definitions and subcontractor management mechanisms, or to not transferring subcontractor technology and knowledge to the program or corporation.

Suitability (1)	Suitability refers to the adequacy with which the selected development model, process, methods, and tools support the scope and type of activities required for the specific program.
Suitability (2)	Suitability of the development system is associated with the degree to which it is supportive of the specific development models, processes, methods, procedures, and activities required and selected for the program. This includes the development, management, documentation, and configuration management processes.
System	The system integration attribute refers to integration of the contractual product to interfacing systems or sites. Factors associated with this attribute are external interface specifications, ability to faithfully produce system interface conditions prior to site or system integration, access to the system or site being interfaced to, adequacy of time for testing, and associate contractor relationships.
System Support	Development system support involves training in use of the system, access to expert users or consultants, and repair or resolution of problems by vendors.
Testability	The testability attribute covers the amenability of the design to testing, design of features to facilitate testing, and the inclusion in the design process of people who will design and conduct product tests.
Type of Contract	This attribute covers the payment terms (cost plus award fee, cost plus fixed fee, etc.) and the contractual requirements associated with such items as the Statement of Work, Contract Data Requirements List, and the amount and conditions of customer involvement.
Unit Test	Factors affecting unit test include planning and preparation and also the resources and time allocated for test. Constituents of these factors are: entering unit test with quality code obtained from formal or informal code inspection or verification procedures; pre-planned test cases

	that have been verified to test unit requirements; a test bed consisting of the necessary hardware or emulators, and software or simulators; test data to satisfy the planned test; and sufficient schedule to plan and carry out the test plan.
Usability	Usability refers to development system documentation, accessibility and workspace, as well as ease of use.
Validity	This attribute refers to whether the aggregate requirements reflect customer intentions for the product. This may be affected by misunderstandings of the written requirements by the contractor or customer, unwritten customer expectations or requirements, or a specification in which the end user did not have inputs. This attribute is affected by the completeness and clarity attributes of the requirements specifications, but refers to the larger question of the system as a whole meeting customer intent.
Vendors	Vendor risks may present themselves in the forms of dependencies on deliveries and support for critical system components.
Work Environment	The work environment element refers to subjective aspects of the environment such as the amount of care given to ensuring that people are kept informed of program goals and information, the way people work together, responsiveness to staff inputs, and the attitude and morale of the program personnel.

11.1.2 Anexo (Tabla N° 2)

Familiarity (1): Familiarity with the development process covers knowledge of, experience in, and comfort with the prescribed process.

En este caso el término Familiarity es un atributo perteneciente al concepto Development Process dentro de la taxonomía de riesgos en el desarrollo de software denominado Development Environment.

Familiarity (2): Development system familiarity depends on prior use of the system by the company and by project personnel as well as adequate training for new users.
En este caso el término Familiarity es un atributo perteneciente al concepto Development System dentro de la taxonomía de riesgos en el desarrollo de software denominado Development Environment.

Feasibility (1): The feasibility attribute refers to the difficulty of implementing a single technical or operational requirement, or of simultaneously meeting conflicting requirements. Sometimes two requirements by themselves are feasible, but together are not; they cannot both exist in the same product at the same time.
Also included is the ability to determine an adequate qualification method for demonstration that the system satisfies the requirement.
En este caso el término Feasibility es un atributo perteneciente al concepto Requirements dentro de la taxonomía de riesgos en el desarrollo de software denominado Product Engineering.

Feasibility (2): The feasibility attribute of the code and unit test element addresses possible difficulties that may arise from poor design or design specification or from inherently difficult implementation needs.
For example, the design may not have quality attributes such as module cohesiveness or interface minimization; the size of the modules may contribute complexity; the design may not be specified in sufficient detail, requiring the programmer to make assumptions or design decisions during coding; or the design and interface specifications may be changing, perhaps without an approved detailed design baseline; and the use of developmental hardware may make an additional contribution to inadequate or unstable interface specification. Or, the nature of the system itself may aggravate the difficulty and complexity of the coding task.
En este caso el término Feasibility es un atributo perteneciente al concepto Code and Unit Test dentro de la taxonomía de riesgos en el desarrollo de software denominado Product Engineering.

Program Interfaces (1): This attribute refers to the interactions of managers at all levels with program personnel at all levels, and with external personnel such as the customer, senior management, and peer managers.
En este caso el término Program Interfaces es un atributo perteneciente al concepto Management Process dentro de la taxonomía de riesgos en el desarrollo de software denominado Development Environment.

Program Interfaces (2): This element consists of the various interfaces with entities and organizations outside the development program itself.
En este caso el término Program Interfaces es un concepto perteneciente a la taxonomía de riesgos en el desarrollo de software denominado Program Constraints.

Reliability (1): System reliability or availability requirements may be affected by hardware not meeting its reliability specifications or system complexity that aggravates difficulties in meeting recovery timelines. Reliability or availability requirements allocated to software may be stated in absolute terms, rather than as separable from hardware and independently testable.
En este caso el término Reliability es un atributo perteneciente al concepto Engineering Specialities dentro de la taxonomía de riesgos en el desarrollo de software denominado Product Engineering.

Reliability (2): Development system reliability is a measure of whether the needed components of the development system are available and working properly whenever required by any program personnel.
En este caso el término Reliability es un atributo perteneciente al concepto Development System dentro de la taxonomía de riesgos en el desarrollo de software denominado Development Environment.

Suitability (1): Suitability refers to the adequacy with which the selected development model, process, methods, and tools support the scope and type of activities required for the specific program.
En este caso el término Suitability es un atributo perteneciente al concepto Development Process dentro de la taxonomía de riesgos en el desarrollo de software denominado Development Environment.

Suitability (2): Suitability of the development system is associated with the degree to which it is supportive of the specific development models, processes, methods, procedures, and activities required and selected for the program. This includes the development, management, documentation, and configuration management processes.
En este caso el término Suitability es un atributo perteneciente al concepto Development System dentro de la taxonomía de riesgos en el desarrollo de software denominado Development Environment.

11.2 Tabla concepto – atributo – valor (TCAV)

En la TCAV se exponen los **conceptos** que forman parte del dominio del sistema y los **atributos** que forman parte de estos conceptos. Los atributos a su vez pueden tomar diferentes **valores** y esto es lo que representa la columna valor de la tabla.
Para tener una idea más clara de los valores que pueden tomar los atributos se adjunta un anexo (ver **Anexo Tabla N° 3 (TCAV)**) al final de la **Tabla N° 3 (TCAV).**

A continuación se muestra la **Tabla N° 3 (TCAV)**:

11.2.1 Tabla N° 3 (TCAV)

Concepto	Atributo	Valor
Requirements	Stability	• Yes • No
	Completeness	• Yes • No
	Clarity	• Yes • No
	Validity	• Yes • No
	Feasibility	• Yes • No
	Precedent	• Yes • No
	Scale	• Yes • No
Design	Funtionality	• Yes • No
	Difficulty	• Yes • No

	Interfaces	• Yes • No
	Performance	• Yes • No
	Testability	• Yes • No
	Harware Constraints	• Yes • No
	Non-Developmental Software	• Yes • No
Code and Unit Test	Feasibility	• Yes • No
	Unit test	• Yes • No
	Coding/Implementation	• Yes • No
Integration and Test	Environment	• Yes • No
	Product	• Yes • No
	System	• Yes • No
Engineering Specialities	Maintainability	• Yes • No
	Reliability	• Yes • No
	Safety	• Yes • No
	Security	• Yes • No
	Human Factors	• Yes • No

	Specifications	• Yes • No
Development Process	Formality	• Yes • No
	Suitability	• Yes • No
	Process Control	• Yes • No
	Familiarity	• Yes • No
	Product Control	• Yes • No
Development System	Capacity	• Yes • No
	Suitability	• Yes • No
	Usability	• Yes • No
	Familiarity	• Yes • No
	Reliability	• Yes • No
	System Support	• Yes • No
	Deliverability	• Yes • No
Management Process	Planning	• Yes • No
	Project Organization	• Yes • No
	Management Experience	• Yes

		• No
	Program Interfaces	• Yes • No
Management Methods	Monitoring	• Yes • No
	Personnel Management	• Yes • No
	Quality Assurance	• Yes • No
	Configuration Management	• Yes • No
Work Environment	Quality Attitude	• Yes • No
	Cooperation	• Yes • No
	Communication	• Yes • No
	Morale	• Yes • No
Resources	Schedule	• Yes • No
	Staff	• Yes • No
	Budget	• Yes • No
	Facilities	• Yes • No
Contract	Type of Contract	• Yes • No
	Restrictions	• Yes • No

	Dependencies	• Yes • No
Program Interfaces	Customer	• Yes • No
	Associate Contractors	• Yes • No
	Subcontractors	• Yes • No
	Prime Contractor	• Yes • No
	Corporate Management	• Yes • No
	Vendors	• Yes • No
	Politics	• Yes • No

11.2.2 Anexo Tabla N° 3 (TCAV)

Este anexo es una herramienta que le permitirá a los usuarios comprender mejor por qué los atributos en la **Tabla N° 3 (TCAV)** toman los valores propuestos.
El orden en el que se expondrán los atributos es el mismo que en el que se encuentran en la **Tabla N° 3 (TCAV)**.

- Stability
  Are requirements changing even as the product is being produced?
- Completeness
  Are requirements missing or incompletely specified?
- Clarity
  Are requirements unclear or in need of interpretation?
- Validity
  Will the requirements lead to the product the customer has in mind?
- Feasibility

Are requirements infeasible from an analytical point of view?

- Precedent

  Do requirements specify something never done before, or that your company has not done before?

- Scale

  Do requirements specify a product larger, more complex, or requiring a larger organization

  Than n the experience of the company?

- Functionality

  Are there any potential problems in meeting functionality requirements?

- Difficulty

  Will the design and/or implementation be difficult to achieve?

- Interfaces

  Are the internal interfaces (hardware and software) well defined and controlled?

- Performance

  Are there stringent response time or throughput requirements?

- Testability

  Is the product difficult or impossible to test?

- Hardware Constraints

  Are there tight constraints on the target hardware?

- Non-Developmental Software

  Are there problems with software used in the program but not developed by the program?

- Feasibility

  Is the implementation of the design difficult or impossible?

- Testing

  Are the specified level and time for unit testing adequate?

- Coding/Implementation

  Are there any problems with coding and implementation?

- Environment

  Is the integration and test environment adequate?

- Product

  Is the interface definition inadequate, facilities inadequate, time insufficient?

- System

  System integration uncoordinated, poor interface definition, or inadequate facilities?

- Maintainability

Will the implementation be difficult to understand or maintain?

- Reliability

  Are the reliability or availability requirements difficult to meet?
- Safety

  Are the safety requirements infeasible and not demonstrable?
- Security

  Are the security requirements more stringent than the current state of the practice or program experience?
- Human Factors

  Will the system will be difficult to use because of poor human interface definition?
- Specifications

  Is the documentation adequate to design, implement, and test the system?
- Formality

  Will the implementation be difficult to understand or maintain?
- Suitability

  Is the process suited to the development model, e.g., spiral, prototyping?
- Process Control

  Is the software development process enforced, monitored, and controlled using metrics? Are distributed development sites coordinated?
- Familiarity

  Are the project members experienced in use of the process? Is the process understood by all staff members?
- Product Control

  Are there mechanisms for controlling changes in the product?
- Capacity

  Is there sufficient work station processing power, memory, or storage capacity?
- Suitability

  Does the development system support all phases, activities, and functions?
- Usability

  How easy is the development system to use?
- Familiarity

  Is there little prior company or project member experience with the development system?
- Reliability

  Does the system suffer from software bugs, down-time, insufficient built-in back-up?
- System Support

Is there timely expert or vendor support for the system?

- Deliverability

  Are the definition and acceptance requirements defined for delivering the development system to the customer

  not budgeted? HINT: If the participants are confused about this, it is probably not an issue from a risk perspective.
- Planning

  Is the planning timely, technical leads included, contingency planning done?
- Project Organization

  Are the roles and reporting relationships clear?
- Management Experience

  Are the managers experienced in software development, software management, the application domain, the development process, or on large programs?
- Program Interfaces

  Is there poor interface with customer, other contractors, senior and/or peer managers?
- Monitoring

  Are management metrics defined and development progress tracked?
- Personnel Management

  Are project personnel trained and used appropriately?
- Quality Assurance

  Are there adequate procedures and resources to assure product quality?
- Configuration Management

  Are the change procedures or version control, including installation site(s), adequate?
- Quality Attitude

  Is there a lack of orientation toward quality work?
- Cooperation

  Is there a lack of team spirit? Does conflict resolution require management intervention?
- Communication

  Is there poor awareness of mission or goals, poor communication of technical information among peers and managers?
- Morale

  Is there a non-productive, non-creative atmosphere? Do people feel that there is no recognition or reward for superior work?
- Schedule

  Is the schedule inadequate or unstable?

- Staff

  Is the staff inexperienced, lacking domain knowledge, lacking skills, or understaffed?
- Budget

  Is the funding insufficient or unstable?
- Facilities

  Are the facilities adequate for building and delivering the product?
- Type of Contract

  Is the contract type a source of risk to the program?
- Restrictions

  Does the contract cause any restrictions?
- Dependencies

  Does the program have any dependencies on outside products or services?
- Customer

  Are there any customer problems such as: lengthy document-approval cycle, poor communication, and inadequate domain expertise?
- Associate Contractors

  Are there any problems with associate contractors such as inadequately defined or unstable interfaces, poor communication, or lack of cooperation?
- Subcontractors

  Is the program dependent on subcontractors for any critical areas?
- Prime Contractor

  Is the program facing difficulties with its Prime contractor?
- Corporate Management

  Is there a lack of support or micro management from upper management?
- Vendors

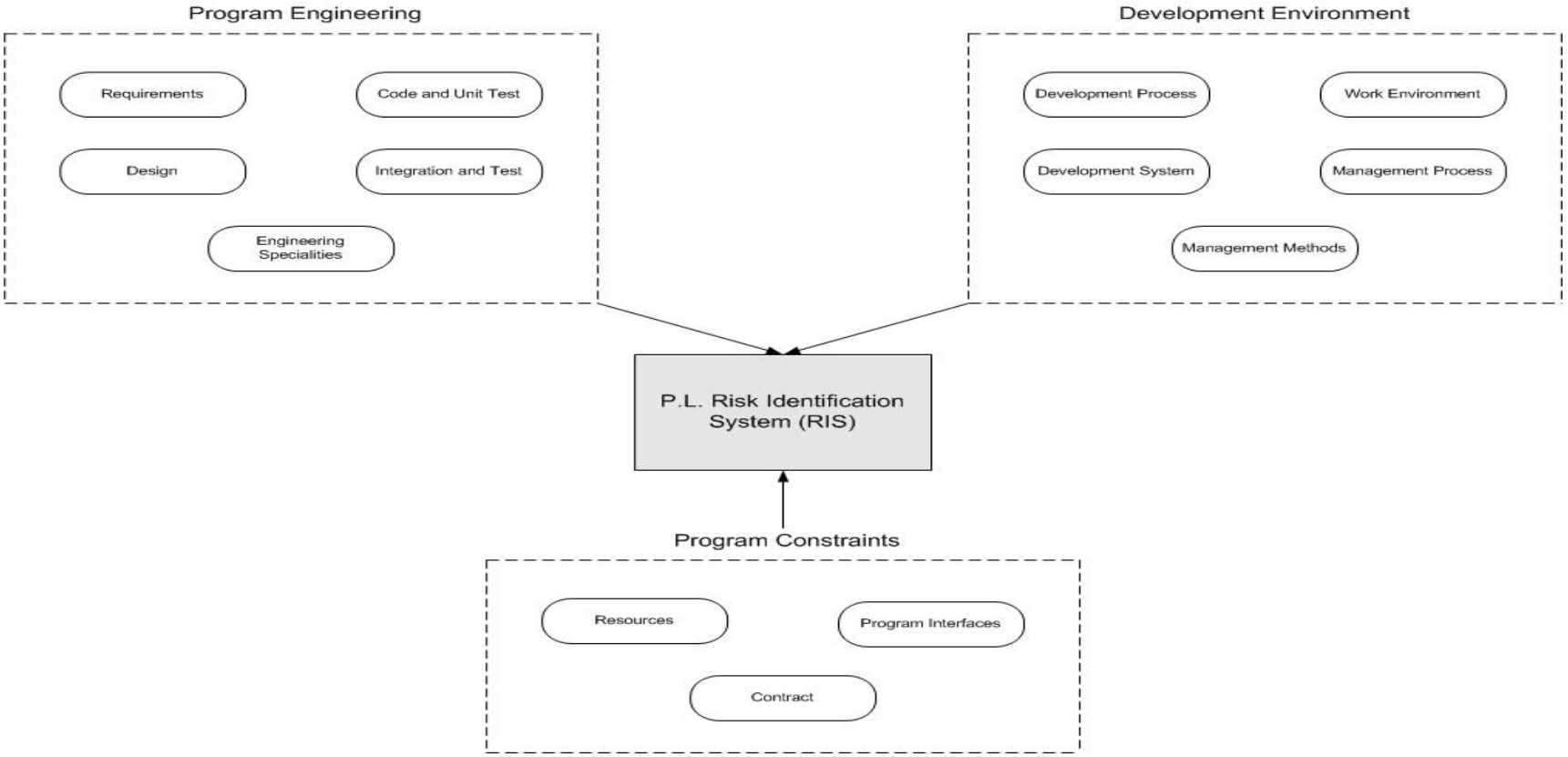
  Are vendors responsive to programs needs?
- Politics

  Are politics causing a problem for the program?

11.3 Mapa de relaciones

En el mapa de relaciones se muestra la relación existente entre los conceptos que forman parte del dominio del sistema. En este caso en paricular nuestro dominio son las distintas taxonomías de riesgo en el desarrollo de software. En la siguiente figura (**Figura N° 1**) se muestra el mapa de relaciones.

Figura N° 1



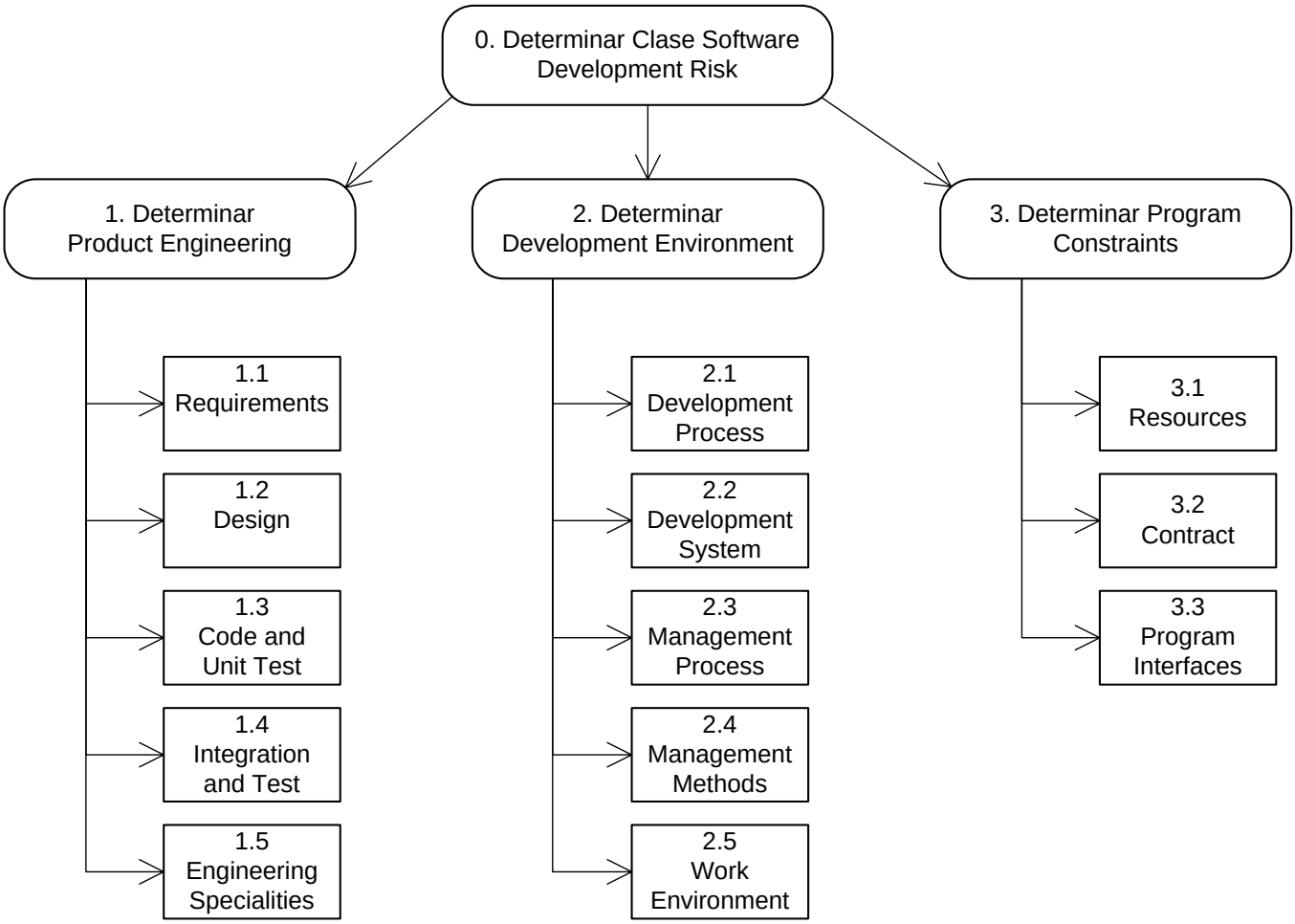
12. Conocimiento Estratégico

En esta sección se desarrolla el tipo de conocimiento relacionado con la manera en que las distintas partes del dominio del sistema experto son aplicadas para la resolución de una tarea. Se especifica qué es lo que hay que hacer, bajo qué condiciones puede hacerse y qué poscondiciones resultarán de lo que se haga; es decir, los conocimientos estratégicos fijan la secuencia de pasos que el Sistema Experto deberá seguir para ejecutar su tarea.

Para ello se utiliza el **Árbol de descomposición funcional**. Este es un árbol invertido, cuya raíz es el objetivo del sistema y cuyas ramas son las estrategias que se llevan a cabo para cumplimentar el objetivo.

Luego de dividir la tarea en pasos modulares, se describen los pasos que componen las hojas del árbol especificando para cada uno: objetivo, precondiciones, entradas, razonamiento y salida.

12.1 Árbol de Descomposición Funcional



12.2 Descripción de Estrategias

Nombre de la estrategia	0. Determinar Clase Software Development Risk
Objetivo	Determinar qué clase de software taxonómico es
Precondiciones	-
Entrada	Elementos de los riesgos y sus atributos
Razonamiento	Característica de cada atributo de los riesgos
Salida	Clase de Software al cual corresponde el riesgo

Nombre de la estrategia	1. Determinar Product Engineering (PE)
Objetivo	Determinar si el riesgo corresponde a PE

Precondiciones	0. Determinar Clase Software Development Risk
Entrada	Elementos de PE y sus atributos
Razonamiento	Características de las actividades realizadas en PE
Salida	El riesgo se manifiesta en PE

Nombre de la estrategia	2. Determinar Development Environment (DE)
Objetivo	Determinar si el riesgo corresponde a DE
Precondiciones	0. Determinar Clase Software Development Risk
Entrada	Elementos de DE y sus atributos
Razonamiento	Características de las actividades realizadas en DE
Salida	El riesgo se manifiesta en DE

Nombre de la estrategia	3. Determinar Program Constraints (PC)
Objetivo	Determinar si el riesgo corresponde a PC
Precondiciones	0. Determinar Clase Software Development Risk
Entrada	Elementos de PC y sus atributos
Razonamiento	Características de las actividades realizadas en PC
Salida	El riesgo se manifiesta en PC

Nombre de la estrategia	1.1 Requirements
Objetivo	Determinar si el riesgo corresponde al elemento Requirements
Precondiciones	Tarea 0 y Tarea 1
Entrada	Atributos de los requerimientos
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Requirements.

Nombre de la estrategia	1.2 Design
Objetivo	Determinar si el riesgo corresponde al elemento Design
Precondiciones	Tarea 0 y Tarea 1
Entrada	Atributos del Diseño
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Design.

Nombre de la estrategia	1.3 Code and Unit Test

Objetivo	Determinar si el riesgo corresponde al elemento Code and Unit Test
Precondiciones	Tarea 0 y Tarea 1
Entrada	Atributos del elemento Code and Unit Test
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Code and Unit Test

Nombre de la estrategia	1.4 Integration and Test
Objetivo	Determinar si el riesgo corresponde al elemento Integration and Test
Precondiciones	Tarea 0 y Tarea 1
Entrada	Atributos del elemento Integration Test
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Integration and Test

Nombre de la estrategia	1.5 Engineering Specialities
Objetivo	Determinar si el riesgo corresponde al elemento Engineering Specialities
Precondiciones	Tarea 0 y Tarea 1
Entrada	Atributos del elemento Engineering Specialities
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Engineering Specialities

Nombre de la estrategia	2.1 Development Process
Objetivo	Determinar si el riesgo corresponde al elemento Development Process
Precondiciones	Tarea 0 y Tarea 2
Entrada	Atributos del elemento Development Process
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Development Process

Nombre de la estrategia	2.2 Development System
Objetivo	Determinar si el riesgo corresponde al elemento Development

	System
Precondiciones	Tarea 0 y Tarea 2
Entrada	Atributos del elemento Development System
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Development System

Nombre de la estrategia	2.3 Management Process
Objetivo	Determinar si el riesgo corresponde al elemento Management Process
Precondiciones	Tarea 0 y Tarea 2
Entrada	Atributos del elemento Management Process
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Management Process

Nombre de la estrategia	2.4 Management Methods
Objetivo	Determinar si el riesgo corresponde al elemento Management Methods
Precondiciones	Tarea 0 y Tarea 2
Entrada	Atributos del elemento Management Methods
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Management Methods

Nombre de la estrategia	2.5 Work Environment
Objetivo	Determinar si el riesgo corresponde al elemento Work Environment
Precondiciones	Tarea 0 y Tarea 2
Entrada	Atributos del elemento Work Environment
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Work Environment

Nombre de la estrategia	3.1 Resources
Objetivo	Determinar si el riesgo corresponde al elemento Resources

Precondiciones	Tarea 0 y Tarea 3
Entrada	Atributos del elemento Resources
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Resources

Nombre de la estrategia	3.2 Contract
Objetivo	Determinar si el riesgo corresponde al elemento Contract
Precondiciones	Tarea 0 y Tarea 3
Entrada	Atributos del elemento Contract
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Contract

Nombre de la estrategia	3.3 Program Interfaces
Objetivo	Determinar si el riesgo corresponde al elemento Program Interfaces
Precondiciones	Tarea 0 y Tarea 3
Entrada	Atributos del elemento Program Interfaces
Razonamiento	Saber que características presenta un riesgo de este tipo
Salida	Manifestación de riesgo correspondiente a Program Interfaces

13. Conocimiento Táctico

Este tipo de conocimiento es el que se refiere a las relaciones que vincula los objetos conceptuales del universo del discurso del dominio de conocimiento del sistema experto.

Lo que se intenta destacar es la causalidad entre conceptos, en particular, de qué modo se pueden inferir los valores de determinados atributos de determinados conceptos a partir de los valores que tienen otros atributos de otros conceptos.

Se especifica cómo y cuando el Sistema Experto puede añadir a sus conocimientos genéricos información actual acerca del caso. Muestran como el SE puede usar corrientemente hechos conocidos.

El conocimiento táctico se modela mediante el uso de reglas y se documenta mediante el uso de Tablas PER (Palabras del Experto-Regla), donde se plantea el cuerpo del conocimiento y las reglas que lo modelan, del tipo SI...ENTONCES.

13.1 Tablas PER

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The definition of what the software product is to do, the needs it must meet, how it is to behave, and how it will be used. This element also addresses the feasibility of developing the product and the scale of the effort.*

Regla	**Si** Stability = "No **ENTONCES** Elemento de Clase Taxonómica = "Requirements-Stability"
Identificador de la Regla	**R1.1 – Requirements-Stability (Product Engineering)**
Regla	**Si** Completeness = "No **ENTONCES** Elemento de Clase Taxonómica = "Requirements-Completeness"
Identificador de la Regla	**R1.2 – Requirements (Product Engineering)**
Regla	**Si** Clarity = "No **ENTONCES** Elemento de Clase Taxonómica = "Requirements-Clarity"
Identificador de la Regla	**R1.3 – Requirements (Product Engineering)**
Regla	**Si** Validity = "No" **ENTONCES** Elemento de Clase Taxonómica = "Requirements-Validity"
Identificador de la Regla	**R1.4 – Requirements (Product Engineering)**
Regla	**Si** Feasibility = "No" **ENTONCES** Elemento de Clase Taxonómica = "Requirements-Feasiblity"
Identificador de la Regla	**R1.5 – Requirements (Product Engineering)**
Regla	**Si** Precedent = "Yes" **ENTONCES**

	Elemento de Clase Taxonómica = “Requirements-Precedent”
Identificador de la Regla	**R1.6 – Requirements (Product Engineering)**
Regla	**Si** Scale = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Requirements-Scale”
Identificador de la Regla	**R1.7 – Requirements (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The translation of requirements into an effective design within project and operational constraints.*
Regla	**Si** Functionality = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Design-Functionality”
Identificador de la regla	**R2.1 – Design (Product Engineering)**
Regla	**Si** Difficulty = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Design-Difficulty”
Identificador de la regla	**R2.2 – Design (Product Engineering)**
Regla	**Si** Interfaces = “No”

	ENTONCES Elemento de Clase Taxonómica = “Design-Interfaces”
Identificador de la regla	**R2.3 – Design (Product Engineering)**
Regla	**Si** Performance = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Design-Performance”
Identificador de la regla	**R2.4 – Design (Product Engineering)**
Regla	**Si** Testability = “No **ENTONCES** Elemento de Clase Taxonómica = “Design-Testability”
Identificador de la regla	**R2.5 – Design (Product Engineering)**
Regla	**Si** Hardware Constraints = Yes” **ENTONCES** Elemento de Clase Taxonómica = “Design-Hardware Constraints”
Identificador de la regla	**R2.6 – Design (Product Engineering)**
Regla	**Si** Non-Developmental Software = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Design- Non-Developmental Software”
Identificador de la regla	**R2.7 – Design (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The translation of software designs into code that satisfies the requirements allocated to individual units.*
Regla	**Si** Feasibility = "No" **ENTONCES** Elemento de Clase Taxonómica = "Code and Unit Test-Feasibility"
Identificador de la regla	**R3.1 – Code and Unit Test (Product Engineering)**
Regla	**Si** Unit Test = "No" **ENTONCES** Elemento de Clase Taxonómica = "Code and Unit Test-Unit Test"
Identificador de la regla	**R3.2 – Code and Unit Test (Product Engineering)**
Regla	**Si** Coding/Implementation = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Code and Unit Test-Coding/Implementation"
Identificador de la regla	**R3.3 – Code and Unit Test (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical*

	activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The integration of units into a working system and the validation that the software product performs as required.
Regla	**Si** Environment = "No" **ENTONCES** Elemento de Clase Taxonómica = "Integration and Test-Environment"
Identificador de la regla	**R4.1 – Integration and Test (Product Engineering)**
Regla	**Si** Product = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Integration and Test-Product"
Identificador de la regla	**R4.2 – Integration and Test (Product Engineering)**
Regla	**Si** System = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Integration and Test-System"
Identificador de la regla	**R4.3 – Integration and Test (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The class focuses on the work to be performed. Product requirements or*

	development activities that may need specialized expertise such as safety, security, and reliability.
Regla	**Si** Maintainability = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Engineering Specialities-Maintainability"
Identificador de la regla	**R5.1 – Engineering Specialities (Product Engineering)**
Regla	**Si** Reliability = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Engineering Specialities-Reliability"
Identificador de la regla	**R5.2 – Engineering Specialities (Product Engineering)**
Regla	**Si** Safety = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Engineering Specialities-Safety"
Identificador de la regla	**R5.3 – Engineering Specialities (Product Engineering)**
Regla	**Si** Security = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Engineering Specialities-Security"
Identificador de la regla	**R5.4 – Engineering Specialities (Product Engineering)**
Regla	**Si** Human Factors = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Engineering Specialities-Human

	Factors”
Identificador de la regla	**R5.5 – Engineering Specialities (Product Engineering)**
Regla	**Si** Specifications = “No” **ENTONCES** Elemento de Clase Taxonómica = “Engineering Specialities-Specifications”
Identificador de la regla	**R5.6 – Engineering Specialities (Product Engineering)**

Estado de la regal	**Texto de la regla**
Palabras del expert	*The development environment class is concerned with the project environment in which a software product is engineered. The definition, planning, documentation, suitability, enforcement, and communication of the methods and procedures used to develop the product.*
Regla	**Si** Formality = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Development Process-Formality”
Identificador de la regla	**R6.1 – Development Process (Development Environment)**
Regla	**SI** Suitability = “No” **ENTONCES** Elemento de Clase Taxonómica = “Development Process-Suitability”
Identificador de la regla	**R6.2 – Development Process (Development Environment)**
Regla	**Si** Process Control = “No”

	ENTONCES Elemento de Clase Taxonómica = “Development Process-Process PControl”
Identificador de la regla	**R6.3 – Development Process (Development Environment)**
Regla	**Si** Familiarity = “No” **ENTONCES** Elemento de Clase Taxonómica = “Development Process-Familiarity”
Identificador de la regla	**R6.4 – Development Process (Development Environment)**
Regla	**Si** Product Control = “No” **ENTONCES** Elemento de Clase Taxonómica = “Development Process- Product Control”
Identificador de la regla	**R6.5 – Development Process (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The tools and supporting equipment used in product development, such as computer-aided software engineering (CASE) tools, simulators, compilers, and host computer systems.*
Regla	**Si** Capacity = “No” **ENTONCES**

	Elemento de Clase Taxonómica = “Development System-Capacity”
Identificador de la Regla	**R7.1 – Development System (Development Environment)**
Regla	**Si** Suitability = “No” **ENTONCES** Elemento de Clase Taxonómica = “Development System-Suitability”
Identificador de la Regla	**R7.2 – Development System (Development Environment)**
Regla	**Si** Usability = “No” **ENTONCES** Elemento de Clase Taxonómica = “Development System-Usability”
Identificador de la Regla	**R7.3 – Development System (Development Environment)**
Regla	**Si** Familiarity = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Development System-Familiarity”
Identificador de la Regla	**R7.4 – Development System (Development Environment)**
Regla	**Si** Reliability = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Development System-Reliability”
Identificador de la Regla	**R7.5 – Development System (Development Environment)**
Regla	**Si** System Support = “No” **ENTONCES** Elemento de Clase Taxonómica = “Development System-System Support”

Identificador de la Regla	**R7.6 – Development System (Development Environment)**
Regla	**Si** Deliverability = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Development System-Deliverability"
Identificador de la Regla	**R7.7 – Development System (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The planning, monitoring, and controlling of budgets and schedules; controlling factors involved in defining, implementing, and testing the product; the project manager's experience in software development, management, and the product domain; and the manager's expertise in dealing with external organizations including customers, senior management, matrix management, and other contractors.*
Regla	**Si** Planning = "No" **ENTONCES** Elemento de Clase Taxonómica = "Management Process-Planning"
Identificador de la regla	**R8.1 – Management Process (Development Environment)**
Regla	**Si** Project Organization = "No" **ENTONCES**

	Elemento de Clase Taxonómica = “Management Process-Project Organization”
Identificador de la regla	**R8.2 – Management Process (Development Environment)**
Regla	**Si** Management Experience = “No” **ENTONCES** Elemento de Clase Taxonómica = “Management Process-Management Experience”
Identificador de la regla	**R8.3 – Management Process (Development Environment)**
Regla	**Si** Program Interfaces = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Management Process-Program Interfaces”
Identificador de la regla	**R8.4 – Management Process (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The methods, tools, and supporting equipment that will be used to manage and control the product development, such as monitoring tools, personnel management, quality assurance, and configuration management.*
Regla	**Si** Monitoring = “No” **ENTONCES**

	Elemento de Clase Taxonómica = “Management Methods-Monitoring”
Identificador de la regla	**R9.1 – Management Methods (Development Environment)**
Regla	**Si** Personnel Management = “No” **ENTONCES** Elemento de Clase Taxonómica = “Management Methods-Personnel Management”
Identificador de la regla	**R9.2 – Management Methods (Development Environment)**
Regla	**Si** Quality Assurance = “No” **ENTONCES** Elemento de Clase Taxonómica = “Management Methods-Quality Assurance”
Identificador de la regla	**R9.3 – Management Methods (Development Environment)**
Regla	**Si** Configuration Management = “No” **ENTONCES** Elemento de Clase Taxonómica = “Management Methods-Configuration Management”
Identificador de la regla	**R9.4 – Management Methods (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The general environment within which the work will be performed, including the*

	attitudes of people and the levels of cooperation, communication, and morale.
Regla	**Si** Quality Attitude = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Work Environment-Quality Attitude"
Identificador de la regla	**R10.1 – Work Environment (Development Environment)**
Regla	**Si** Cooperation = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Work Environment-Cooperation"
Identificador de la regla	**R10.2 – Work Environment (Development Environment)**
Regla	**Si** Communication = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Work Environment-Communication"
Identificador de la regla	**R10.3 – Work Environment (Development Environment)**
Regla	**Si** Morale = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Work Environment-Morale"
Identificador de la regla	**R10.4 – Work Environment (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but*

	can still have major effects on its success. The external constraints imposed on schedule, staff, budget, or facilities.
Regla	**Si** Schedule = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Resources-Schedule"
Identificador de la regla	**R11.1 – Resources (Program Constraints)**
Regla	**Si** Staff = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Staff"
Identificador de la regla	**R11.2 – Resources (Program Constraints)**
Regla	**Si** Budget = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Resources-Budget"
Identificador de la regla	**R11.3 – Resources (Program Constraints)**
Regla	**Si** Facilities = "No" **ENTONCES** Elemento de Clase Taxonómica = "Resources-Facilities"
Identificador de la regla	**R11.4 – Resources (Program Constraints)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but*

	can still have major effects on its success. The terms and conditions of the project contract.
Regla	**Si** Type of Contract = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Contract-Type of Contract"
Identificador de la regla	**R12.1 – Contract (Program Constraints)**
Regla	**Si** Restrictions = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Contract-Restrictions"
Identificador de la regla	**R12.2 – Contract (Program Constraints)**
Regla	**Si** Dependencies = "Yes" **ENTONCES** Elemento de Clase Taxonómica = "Contract-Dependencies"
Identificador de la regla	**R12.3 – Contract (Program Constraints)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but can still have major effects on its success. The external interfaces to customers, other contractors, corporate management, and vendors.*
Regla	**Si** Customer = "Yes" **ENTONCES**

	Elemento de Clase Taxonómica = “Program Interfaces-Customer”
Identificador de la regla	**R13.1 – Program Interfaces (Program Constraints)**
Regla	**Si** Associate Contractors = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Program Interfaces-Associate Contractors”
Identificador de la regla	**R13.2 – Program Interfaces (Program Constraints)**
Regla	**Si** Subcontractors = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Program Interfaces-Subcontractors”
Identificador de la regla	**R13.3 – Program Interfaces (Program Constraints)**
Regla	**Si** Prime Contractor = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Program Interfaces-Prime Contractor”
Identificador de la regla	**R13 .4– Program Interfaces (Program Constraints)**
Regla	**Si** Corporate Management = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Program Interfaces-Corporate Management”
Identificador de la regla	**R13.5 – Program Interfaces (Program Constraints)**
Regla	**Si** Vendors = “No”

	ENTONCES Elemento de Clase Taxonómica = “Program Interfaces-Vendors”
Identificador de la regla	**R13.6 – Program Interfaces (Program Constraints)**
Regla	**Si** Politics = “Yes” **ENTONCES** Elemento de Clase Taxonómica = “Program Interfaces-Politics”
Identificador de la regla	**R13.7 – Program Interfaces (Program Constraints)**

13.2 Reglas para conclusiones

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The definition of what the software product is to do, the needs it must meet, how it is to behave, and how it will be used. This element also addresses the feasibility of developing the product and the scale of the effort.*
Regla	**Si** Stability = “No” **O** Completeness = “No” **O** Clarity = “No” **O** Validity = “No”

	O Feasibility = “No” **O** Precedent = “Yes” **O** Scale = “Yes” **ENTONCES** Requirements-no-conclusion = “FALSE” Elemento de Clase Taxonómica = “Requirements”
Nombre de la regla	**R1 – Requirements (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The translation of requirements into an effective design within project and operational constraints.*
Formulación externa	**Si** Functionality = “Yes” **O** Difficulty = “Yes” **O** Interfaces = “No” **O** Performance = “Yes” **O** Testability = “No” **O** Hardware Constraints = Yes” **O** Non-Developmental Software = “Yes”

	ENTONCES Design-no-conclusion = “FALSE” Elemento de Clase Taxonómica = “Design”
Nombre de la regla	**R2 – Design (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The translation of software designs into code that satisfies the requirements allocated to individual units.*
Formulación externa	**Si** Feasibility = “No” **O** Unit Test = “No” **O** Coding/Implementation = “Yes” **ENTONCES** Test-no-conclusion = “FALSE” Elemento de Clase Taxonómica = “Code and Unit Test”
Nombre de la regla	**R3 – Code and Unit Test (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical*

	activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The integration of units into a working system and the validation that the software product performs as required.
Formulación externa	**Si** Environment = "No" **O** Product = "Yes" **O** System = "Yes" **ENTONCES** Integration-no-conclusion = "FALSE" Elemento de Clase Taxonómica = "Integration and Test"
Nombre de la regla	**R4 – Integration and Test (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The class focuses on the work to be performed. Product requirements or development activities that may need specialized expertise such as safety, security, and reliability.*
Formulación externa	**Si** Maintainability = "Yes" **O** Reliability = "Yes" **O** Safety = "Yes"

	O Security = “Yes” **O** Human Factors = “Yes” **O** Specifications = “No” **ENTONCES** Engineering-no-conclusion = “FALSE” Elemento de Clase Taxonómica = “Engineering Specialities”
Nombre de la regla	**R5 – Engineering Specialities (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The definition, planning, documentation, suitability, enforcement, and communication of the methods and procedures used to develop the product.*
Formulación externa	**Si** Formality = “Yes” **O** Suitability = “No” **O** Process Control = “No” **O** Familiarity = “No” **O** Product Control = “No” **ENTONCES** Dev Proc-no-conclusion = “FALSE” Elemento de Clase Taxonómica = “Development Process”
Nombre de la regla	**R6 – Development Process (Development Environment)**

Estado de la regla	Texto de la regla
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The tools and supporting equipment used in product development, such as computer-aided software engineering (CASE) tools, simulators, compilers, and host computer systems.*
Formulación externa	**Si** Capacity = "No" **O** Suitability = "No" **O** Usability = "No" **O** Familiarity = "Yes" **O** Reliability = "Yes" **O** System Support = "No" **O** Deliverability = "Yes" **ENTONCES** Dev Sys-no-conclusion = "FALSE" Elemento de Clase Taxonómica = "Development System"
Nombre de la Regla	**R7 – Development System (Development Environment)**

Estado de la regla	Texto de la regla
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The planning,*

	monitoring, and controlling of budgets and schedules; controlling factors involved in defining, implementing, and testing the product; the project manager's experience in software development, management, and the product domain; and the manager's expertise in dealing with external organizations including customers, senior management, matrix management, and other contractors.
Formulación externa	**Si** Planning = "No" **O** Project Organization = "No" **O** Management Experience = "No" **O** Program Interfaces = "Yes" **ENTONCES** Man Proc-no-conclusion = "FALSE" Elemento de Clase Taxonómica = "Management Process"
Nombre de la regla	**R8 – Management Process (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The methods, tools, and supporting equipment that will be used to manage and control the product development, such as monitoring tools, personnel management, quality assurance, and configuration management.*
Formulación externa	**Si** Monitoring = "No"

	O Personnel Management = "No" **O** Quality Assurance = "No" **O** Configuration Management = "No" **ENTONCES** Man Met-no-conclusion = "FALSE" Elemento de Clase Taxonómica = "Management Methods"
Nombre de la regla	**R9 – Management Methods (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The general environment within which the work will be performed, including the attitudes of people and the levels of cooperation, communication, and morale.*
Formulación externa	**Si** Quality Attitude = "Yes" **O** Cooperation = "Yes" **O** Communication = "Yes" **O** Morale = "Yes" **ENTONCES** Work Env-no-conclusion = "FALSE" Elemento de Clase Taxonómica = "Work Environment"
Nombre de la regla	**R10 – Work Environment (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but can still have major effects on its success. The external constraints imposed on schedule, staff, budget, or facilities.*
Formulación externa	**Si** Schedule = "Yes" **O** Staff = "Yes" **O** Budget = "Yes" **O** Facilities = "No" **ENTONCES** Resources-no-conclusion = "FALSE" Elemento de Clase Taxonómica = "Resources"
Nombre de la regla	**R11 – Resources (Program Constraints)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but can still have major effects on its success. The terms and conditions of the project contract.*
Formulación externa	**Si** Type of Contract = "Yes" **O** Restrictions = "Yes"

	O Dependencies = “Yes” **ENTONCES** Contract-no-conclusion = “FALSE” Elemento de Clase Taxonómica = “Contract”
Nombre de la regla	**R12 – Contract (Program Constraints)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The program constraints class consists of the “externals” of the project—the factors that are outside the direct control of the project but can still have major effects on its success. The external interfaces to customers, other contractors, corporate management, and vendors.*
Formulación externa	**Si** Customer = “Yes” **O** Associate Contractors = “Yes” **O** Subcontractors = “Yes” **O** Prime Contractor = “Yes” **O** Corporate Management = “Yes” **O** Vendors = “No” **O** Politics = “Yes” **ENTONCES** Interfaces-no-conclusion = “FALSE” Elemento de Clase Taxonómica = “Program Interfaces”
Nombre de la regla	**R13 – Program Interfaces (Program Constraints)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*If there are no risks in the previous elements, then there are no risks in this system.*
Formulación externa	**Si** Requirements-no-conclusion = “TRUE” **O** Design-no-conclusion = “TRUE” **O** Test-no-conclusion = “TRUE” **O** Integration-no-conclusion = “TRUE” **O** Engineering-no-conclusion = “TRUE” **O** Dev Proc-no-conclusion = “TRUE” **O** Dev Sys-no-conclusion = “TRUE” **O** Man Proc-no-conclusion = “TRUE” **O** Man Met-no-conclusion = “TRUE” **O** Work Env-no-conclusion = “TRUE” **O** Resources-no-conclusion = “TRUE” **O** Contract-no-conclusion = “TRUE” **O** Interfaces-no-conclusion = “TRUE” **ENTONCES** Conclusion = “No risks were found”
Nombre de la regla	**R14 – No Risk**

14. Módelo Dinámico

14.1 Introducción

Consiste en la integración de los conocimientos fácticos, estratégicos y tácticos. El modelo dinámico es un modelo que permite comprobar en definitiva si no existen demasiados errores ni olvidos en el proceso que realiza el experto.

Se utilizan dos técnicas: el Árbol de jerarquía de tareas, armado con árbol de descomposición funcional, la T.C.A.V. y las seudoreglas; y el Mapa de conocimientos, que representa el proceso de inferir valores de los atributos y los enlaces entre los atributos.

14.2 Mapa de Conocimientos

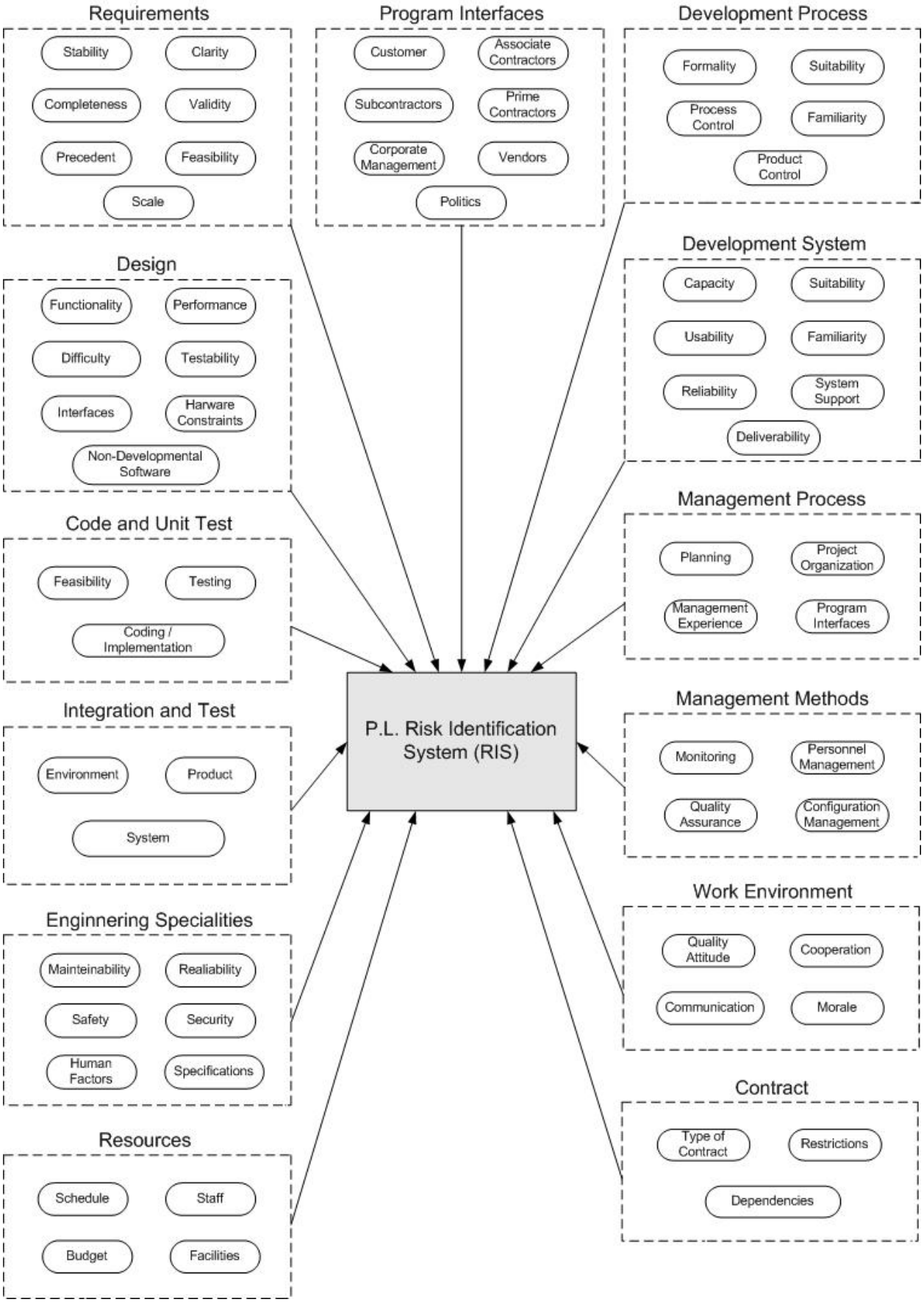
En la página actual y la siguiente se representa el mapa de conocimientos que se ha logrado como resultado del análisis de conocimientos fácticos realizado. El mapa de conocimientos en definitiva refleja cómo está ubicado el conocimiento del experto.

En cada nodo del mapa de conocimientos se pueden destacar los atributos que pertenecen al dominio del sistema.

Los atributos fueron encapsulados dentro del concepto al cual pertenecen para que se comprenda la idea global que se intenta transmitir.

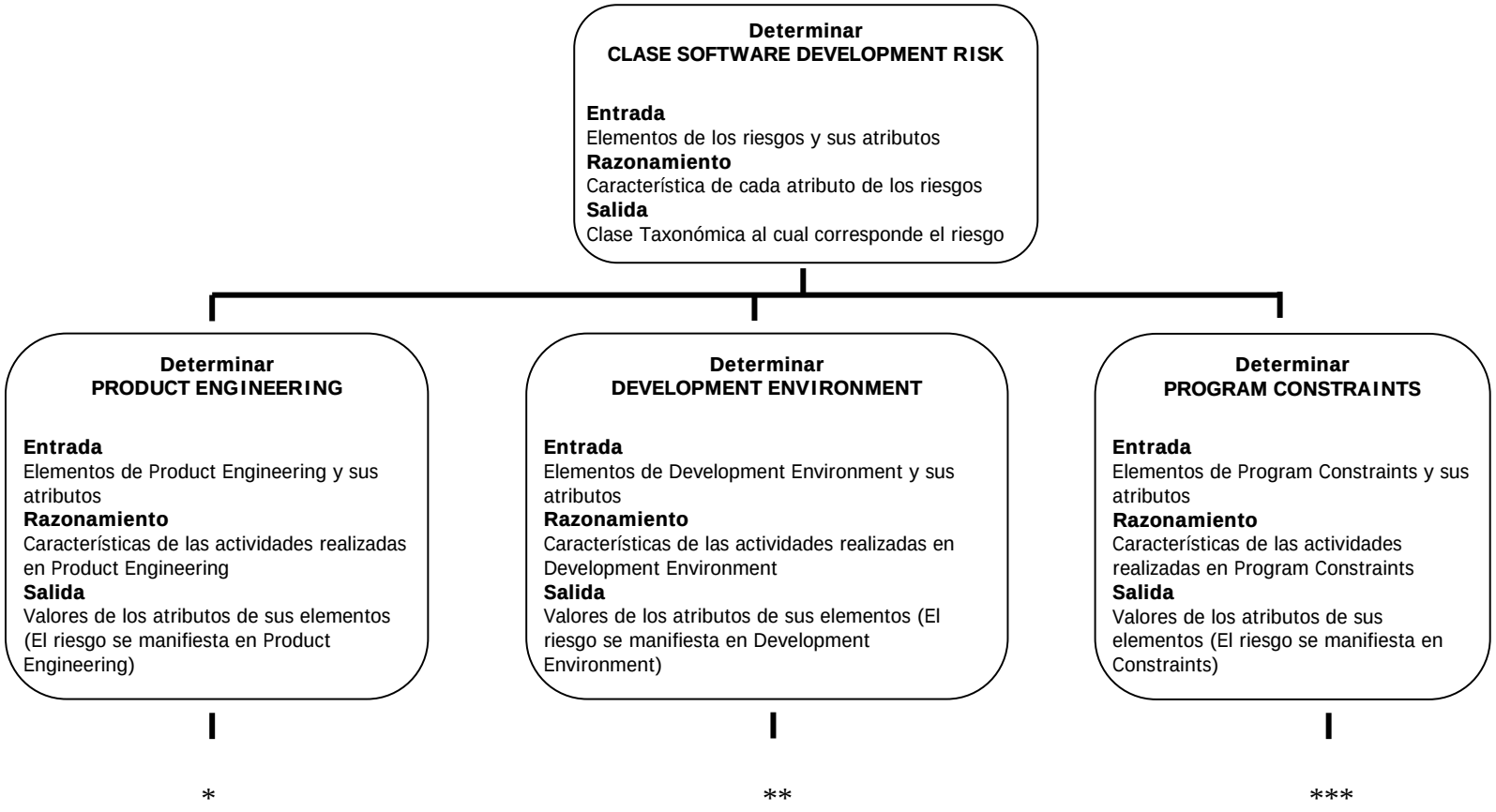
En la **Figura N° 2** se presenta el mapa de conocimientos.

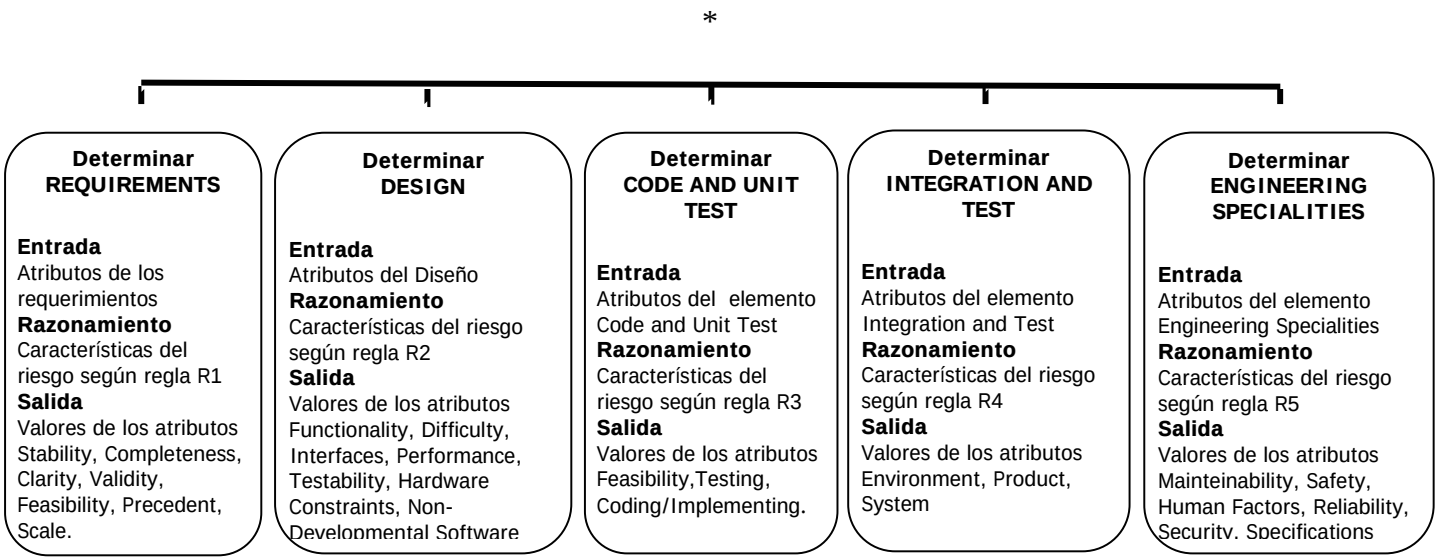
Figura N° 2

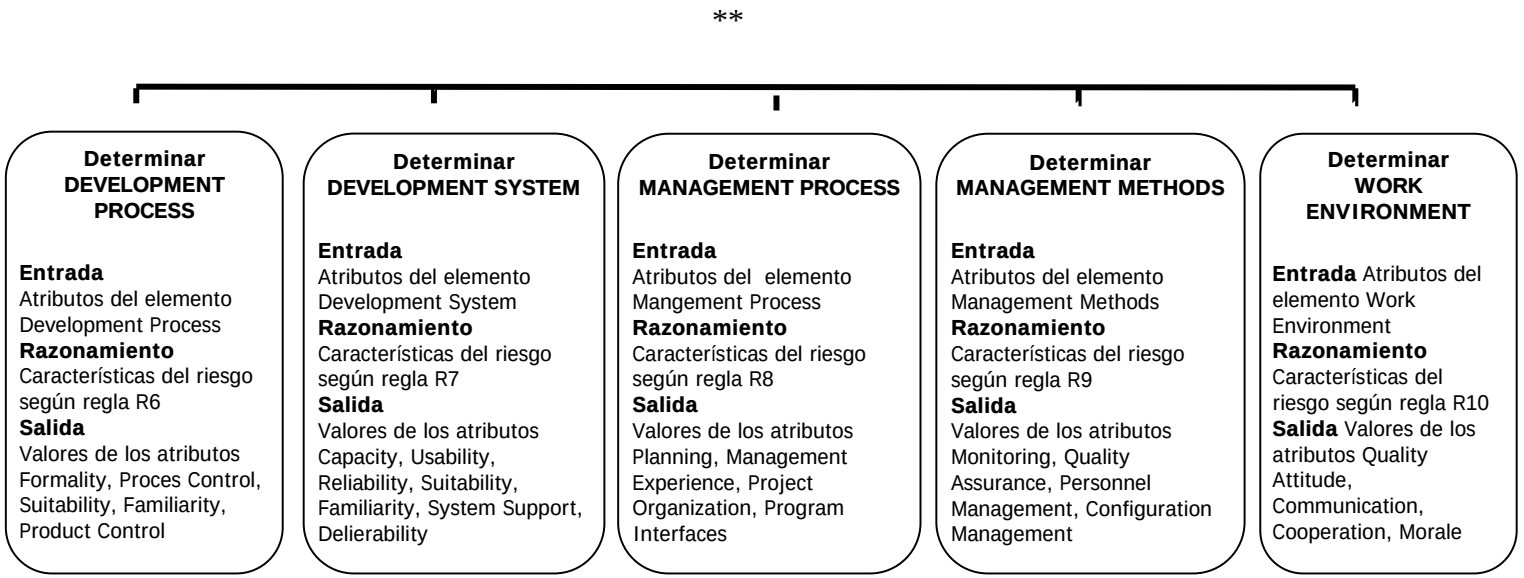

Requirements
Stability
Clarity
Completeness
Validity
Precedent
Feasibility
Scale
Program Interfaces
Customer
Associate Contractors
Subcontractors
Prime Contractors
Corporate Management
Vendors
Politics
Development Process
Formality
Suitability
Process Control
Familiarity
Product Control
Design
Functionality
Performance
Difficulty
Testability
Interfaces
Harware Constraints
Non-Developmental Software
Development System
Capacity
Suitability
Usability
Familiarity
Reliability
System Support
Deliverability
Code and Unit Test
Feasibility
Testing
Coding / Implementation
Management Process
Planning
Project Organization
Management Experience
Program Interfaces
P.L. Risk Identification System (RIS)
Integration and Test
Environment
Product
System
Management Methods
Monitoring
Personnel Management
Quality Assurance
Configuration Management
Work Environment
Quality Attitude
Cooperation
Communication
Morale
Enginnering Specialities
Mainteinability
Realiability
Safety
Security
Human Factors
Specifications
Contract
Type of Contract
Restrictions
Dependencies
Resources
Schedule
Staff
Budget
Facilities

14.3 Árbol Jerárquico

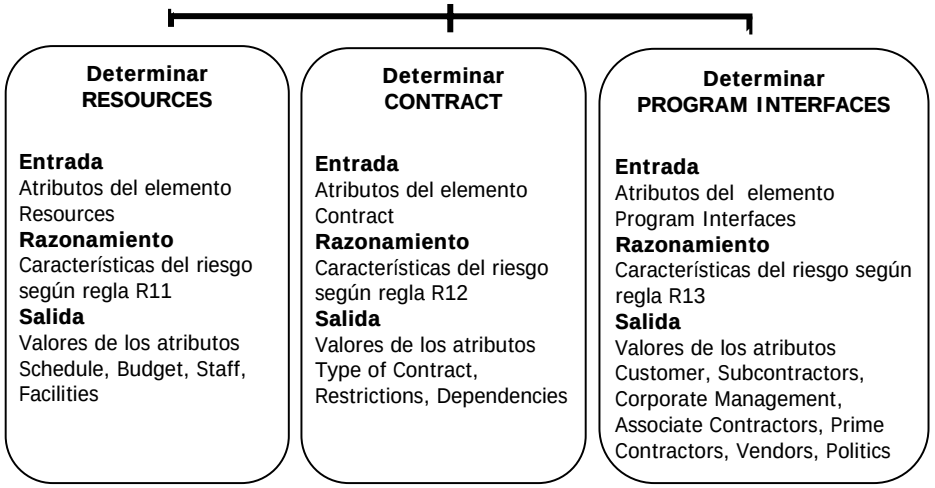
- En cada nodo del árbol jerárquico que se corresponde con el árbol de descomposición funcional de los conocimientos estratégicos se puede destacar:
  - Las entradas, salidas y conceptos se corresponden a los conceptos y atributos definidos en la tabla conceptos - atributos anteriormente definidos por el grupo en el trabajo de conocimientos fácticos.
  - Mientras que el razonamiento se corresponde a las reglas y fórmulas definidas en los conocimientos tácticos.


Determinar
CLASE SOFTWARE DEVELOPMENT RISK
Entrada
Elementos de los riesgos y sus atributos
Razonamiento
Característica de cada atributo de los riesgos
Salida
Clase Taxonómica al cual corresponde el riesgo
Determinar
PRODUCT ENGINEERING
Entrada
Elementos de Product Engineering y sus atributos
Razonamiento
Características de las actividades realizadas en Product Engineering
Salida
Valores de los atributos de sus elementos (El riesgo se manifiesta en Product Engineering)
Determinar
DEVELOPMENT ENVIRONMENT
Entrada
Elementos de Development Environment y sus atributos
Razonamiento
Características de las actividades realizadas en Development Environment
Salida
Valores de los atributos de sus elementos (El riesgo se manifiesta en Development Environment)
Determinar
PROGRAM CONSTRAINTS
Entrada
Elementos de Program Constraints y sus atributos
Razonamiento
Características de las actividades realizadas en Program Constraints
Salida
Valores de los atributos de sus elementos (El riesgo se manifiesta en Constraints)
*
**
***


*
Determinar REQUIREMENTS
Entrada
Atributos de los requerimientos
Razonamiento
Características del riesgo según regla R1
Salida
Valores de los atributos Stability, Completeness, Clarity, Validity, Feasibility, Precedent, Scale.
Determinar DESIGN
Entrada
Atributos del Diseño
Razonamiento
Características del riesgo según regla R2
Salida
Valores de los atributos Functionality, Difficulty, Interfaces, Performance, Testability, Hardware Constraints, Non-
Determinar CODE AND UNIT TEST
Entrada
Atributos del elemento Code and Unit Test
Razonamiento
Características del riesgo según regla R3
Salida
Valores de los atributos Feasibility,Testing, Coding/Implementing.
Determinar INTEGRATION AND TEST
Entrada
Atributos del elemento Integration and Test
Razonamiento
Características del riesgo según regla R4
Salida
Valores de los atributos Environment, Product, System
Determinar ENGINEERING SPECIALITIES
Entrada
Atributos del elemento Engineering Specialities
Razonamiento
Características del riesgo según regla R5
Salida
Valores de los atributos Mainteinability, Safety, Human Factors, Reliability, Security, Specifications


**
Determinar DEVELOPMENT PROCESS
Entrada
Atributos del elemento Development Process
Razonamiento
Características del riesgo según regla R6
Salida
Valores de los atributos Formality, Proces Control, Suitability, Familiarity, Product Control
Determinar DEVELOPMENT SYSTEM
Entrada
Atributos del elemento Development System
Razonamiento
Características del riesgo según regla R7
Salida
Valores de los atributos Capacity, Usability, Reliability, Suitability, Familiarity, System Support, Delierability
Determinar MANAGEMENT PROCESS
Entrada
Atributos del elemento Mangement Process
Razonamiento
Características del riesgo según regla R8
Salida
Valores de los atributos Planning, Management Experience, Project Organization, Program Interfaces
Determinar MANAGEMENT METHODS
Entrada
Atributos del elemento Management Methods
Razonamiento
Características del riesgo según regla R9
Salida
Valores de los atributos Monitoring, Quality Assurance, Personnel Management, Configuration Management
Determinar WORK ENVIRONMENT
Entrada Atributos del elemento Work Environment
Razonamiento
Características del riesgo según regla R10
Salida Valores de los atributos Quality Attitude, Communication, Cooperation, Morale

***


Determinar RESOURCES
Entrada
Atributos del elemento Resources
Razonamiento
Características del riesgo según regla R11
Salida
Valores de los atributos Schedule, Budget, Staff, Facilities
Determinar CONTRACT
Entrada
Atributos del elemento Contract
Razonamiento
Características del riesgo según regla R12
Salida
Valores de los atributos Type of Contract, Restrictions, Dependencies
Determinar PROGRAM INTERFACES
Entrada
Atributos del elemento Program Interfaces
Razonamiento
Características del riesgo según regla R13
Salida
Valores de los atributos Customer, Subcontractors, Corporate Management, Associate Contractors, Prime Contractors, Vendors, Politics

15. Formalización

15.1 Introducción

Luego de la conceptualización se procede a expresar dichos conocimientos de una manera formal. La etapa de formalización tiene como objetivo expresar los conocimientos sobre el problema y su resolución en estructuras que puedan ser utilizadas por una computadora.

15.2 Marcos

15.2.1 Marco clase PL RIS

MC PL RIS	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Nombre	Conjunto de caracteres	1/1	No	-	P.L. Risk Identification System	-	-
Empresa	Conjunto de caracteres	1/1	No	-	P.L.	-	-
Fecha	Date	1/1	No	-	2008	-	-

15.2.2 Marco clase Requirements

MC Requirements	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Stability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Completeness	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Clarity	Conjunto de caracteres	1/1	No	-	Yes, no	-	-

Validity	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Feasibility	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Precedent	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Scale	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.3 Marco clase Design

MC Design	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Funtionality	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Difficulty	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Interfaces	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Performance	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Testability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Hardware Constraints	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.4 Marco clase Code and Unit Test

MC Code and Unit Test	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Feasibility	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Unit test	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Coding / Implementation	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.5 Marco clase Integration and Test

MC Integration and Test	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Environment	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Product	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
System	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.6 Marco clase Engineering Specialities

MC Engineering Specialities	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Maintainability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Reliability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Safety	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Human Factors	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Specifications	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.7 Marco clase Development Process

MC Development Process	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Formality	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Suitability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Process Control	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Familiarity	Conjunto de caracteres	1/1	No	-	Yes, no	-	-

Product Control	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.8 Marco clase Development System

MC Development System	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Capacity	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Suitability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Usability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Familiarity	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Reliability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
System Support	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Deliverability	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.9 Marco clase Management Process

MC Management Process	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Planning	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Project Organization	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Management Experience	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Program Interfaces	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.10 Marco clase Management Methods

MC Management Methods	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Monitoring	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Personnel Management	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Quality Assurance	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Configuration Management	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.11 Marco clase Work Environment

MC Work Environment	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Quality Attitude	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Cooperation	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Communication	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Morale	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.12 Marco clase Resources

MC Resources	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Schedule	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Staff	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Budget	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Facilities	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.13 Marco clase Contract

MC Contract	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Type of Contract	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Restrictions	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Dependencies	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.2.14 Marco clase Program Interfaces

MC Program Interfaces	Tipo Ranura	Modal. / Cardin.	Multiv.	Prop. General	Valores Permit.	Valores omisión	Si Necesito
Customer	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Associate Contractors	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Subcontractors	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Prime Contractor	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Corporate Management	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Vendors	Conjunto de caracteres	1/1	No	-	Yes, no	-	-
Politics	Conjunto de	1/1	No	-	Yes, no	-	-

	caracteres						
Sub-clase de	Marco	1/1	No	MC PL RIS	-	-	-

15.3 Reglas de Producción

Las reglas de producción estarán dadas por las seudo reglas del Conocimiento Táctico llevadas a algún lenguaje de programación que permita llevar a cabo el sistema experto.

16. Referencias

Alonso de la Barra, A., Haddad, R., Herrera, C. (1998). Sistema Experto de Asistencia Técnica de Automóviles. Universidad de Santiago de Chile Facultad de Ciencias.

Apache Software Foundation. (2010). Apache Struts. http://struts.apache.org. Vigente al 01/03/2010.

Bloch, J. (2008). Effective Java, 2nd Edition. Prentice Hall. ISBN 978-0321356680.

Brown, D., Davis, C., Stanlick, S. (2008). Struts 2 in Action. Manning. ISBN 978-1933988078.

Carr, M., Konda, S., Monarch, I., Ulrich, C., Walker, C. (1993). Taxonomy-Based Risk Identification. Technical Report CMU/SEI-93-TR-6 ESC-TR-93-183.

Clips. (2010). Clips. http://clipsrules.sourceforge.net/. Vigente al 01/03/2010.

Degl'Innocenti, A., Salcedo, J., Hollman, E., Britos, P., García-Martínez, R., Rossi, B. (2000). Sistema Experto en Análisis de Fallas en Líneas Eléctricas de Transmisión. Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS). ITBA.

Eckel, B. (2006). Thinking in Java, 4th Edition. Prentice Hall. ISBN 978-0131872486.

Freeman, E., Freeman, E. (2005). Head First HTML with CSS & XHTML. O'Reilly Media. ISBN 978-0596101978.

García Martínez, R. (1997). Sistemas Autónomos. Nueva Librería. ISBN 950-9088-84-6

García Martinez, R., Britos ,P. (2004). Ingeniería de Sistemas Expertos. Nueva Librería. ISBN 987-1104-15-4.

García Martinez, R., Servente, M., Pasquini, D. (2003). Sistemas Inteligentes. Nueva Librería. ISBN 987-1104-05-7.

Gomez, S., Perichinsky, G. y García-Martínez, R. (2001). Argumentación Utilizando Razonamiento Basado en Precedentes en Sistemas Expertos Legales. ISBN 950-31-0050-X.

Hall, M., Brown, L. (2004). Core Servelets and JavaServer Pages, 2nd Edition. Prentice Hall. ISBN 978-0130092298.

Lightbody, P., Carreira, J. (2008). Webwork in Action. Manning. ISBN 978-1932394535.

Liguori, R., Liguori, P. (2008). Java Pocket Guide. O'Reilly Media. ISBN 978-0596514198.

Montes, J. (2009). Sistemas Expertos. http://www.monografias.com/trabajos16/sistemas-expertos/sistemas-expertos.shtml. Vigente al 01/03/2010.

Musciano, C., Kennedy, B. (2006). HTML & XHTML: The Definitive Guide, Sixth Edition. O'Reilly Media. ISBN 978-0596003821.

W3Schools. (2010). HTML Tutorial. http://www.w3schools.com/html/. Vigente al 01/03/2010.

Wikipedia. (2010). Diagrama de Gantt. http://es.wikipedia.org/wiki/Diagrama_de_Gantt. Vigente al 01/03/2010.

Wikipedia. (2010). Modelo Vista Controlador. http://es.wikipedia.org/wiki/Modelo_Vista_Controlador. Vigente al 01/03/2010.

Wikipedia. (2010). Sistema Experto. http://es.wikipedia.org/wiki/Sistema_experto. Vigente al 01/03/2010.

17. Anexo I – Manual de Usabilidad

17.1 Introducción

El presente trabajo tiene como objetivo plasmar mediante una interfaz grafica amigable para el usuario un sistema experto de identificación de riesgos en desarrollos de software.

Los efectos de la introducción del sistema experto se prevé que ayudara a encontrar los riesgos en tiempos tempranos para, entre otras cosas, evitar el uso ineficiente de recursos humanos. Se espera que el cuestionario vaya variando en el tiempo a conveniencia de los usuarios y para mejorar los fines que apunta, por lo tanto se dice que es un sistema vivo que con pequeñas modificaciones en el motor de inferencia y en la capa de presentación se adecua y progresa (acepta la técnica de prototipado gradual).

La ubicación idónea para este sistema experto es o bien una universidad o una empresa dedicada al desarrollo de software. La identificación temprana de riesgos es de vital importancia para la subsistencia de una organización.
Se considera al sistema como una herramienta de apoyo que no interfiere en el trabajo cotidiano de los empleados aunque puede ser utilizado para la toma de decisiones por parte de la gerencia o el líder del proyecto informático. El nivel de formación requerido por los usuarios del sistema debe incluir conocimientos en informática para poder analizar, entender y responder las preguntas del cuestionario. Se debe contar con poco hardware y software para la utilización del sistema, siendo muy poco restrictivo al respecto.

El problema de identificación de riesgos se puede dividir en subproblemas.

- **Ingeniería de producto**
    - Requerimientos
    - Diseño
    - Código y unidad de prueba
    - Integración y testeo
    - Especialidades de ingeniería
- **Ambiente de desarrollo**
    - Proceso de desarrollo
    - Sistema de desarrollo
    - Proceso de gestión

- Métodos de gestión
- Ambiente laboral

- **Limitaciones de programa**
  - Recursos
  - Contratos
  - Interfaces de programa

Por cada subproblema, se estudia su clase con sus atributos asociados y los valores que éstos pueden tomar. De las respuestas del cuestionario se infiere los riegos enfrentados actualmente.

17.2 SE – Sistema Experto

17.2.1 Motor de inferencia

Fue desarrollado en el lenguaje CLIPS. Originalmente el mismo incluía todo el cuestionario TBQ (Taxonomy-Base questionnaire) definido como reglas. Para esta versión, esas reglas fueron eliminadas, dejando solamente las que verdaderamente sirven para detectar los riesgos y que se presentan en el siguiente ítem.

17.2.2 Reglas

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The definition of what the software product is to do, the needs it must meet, how it is to behave, and how it will be used. This element also addresses the feasibility of developing the product and the scale of the effort.*
Regla	**Si** Stability = “yes” **y** Completeness = “no” **y** Clarity = “no” **y** Validity = “yes” **y** Feasibility = “no” **y** Precedent = “no” **y** Scale = “no” **ENTONCES** Requirements-no-conclusion = “TRUE” Elemento de Clase Taxonómica = “Requirements”
Nombre de la regla	**R1 – Requirements (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The translation of requirements into an effective design within project and operational constraints.*
Formulación externa	**Si** Functionality = "no" **y** Difficulty = "no" **y** Interfaces = "yes" **y** Performance = "no" **y** Testability = "no" **y** Hardware Constraints = "no" **y** Non-Developmental Software = "no" **ENTONCES** Design-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Design"
Nombre de la regla	**R2 – Design (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The translation of software designs into code that satisfies the requirements allocated to individual units.*
Formulación externa	**Si** Feasibility = "no" **y** Unit Test = "yes" **y** Coding/Implementation = "no" **ENTONCES** Test-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Code and Unit Test"
Nombre de la regla	**R3 – Code and Unit Test (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The integration of units into a working system and the validation that the software product performs as required.*

Formulación externa	**Si** Environment = “yes” **y** Product = “no” **y** System = “no” **ENTONCES** Integration-no-conclusion = “TRUE” Elemento de Clase Taxonómica = “Integration and Test”
Nombre de la regla	**R4 – Integration and Test (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The product engineering class consists of the intellectual and physical activities required to build the product to be delivered to the customer. It includes the complete system hardware, software, and documentation. The class focuses on the work to be performed. Product requirements or development activities that may need specialized expertise such as safety, security, and reliability.*
Formulación externa	**Si** Maintainability = “no” **y** Reliability = “no” **y** Safety = “no” **y** Security = “no” **y** Human Factors = “no” **y** Specifications = “yes” **ENTONCES** Engineering-no-conclusion = “TRUE” Elemento de Clase Taxonómica = “Engineering Specialities”
Nombre de la regla	**R5 – Engineering Specialities (Product Engineering)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The definition, planning, documentation, suitability, enforcement, and communication of the methods and procedures used to develop the product.*
Formulación externa	**Si** Formality = “no” **y** Suitability = “yes” **y** Process Control = “yes” **y** Familiarity = “yes” **y** Product Control = “yes” **ENTONCES** Dev Proc-no-conclusion = “TRUE” Elemento de Clase Taxonómica = “Development Process”
Nombre de la regla	**R6 – Development Process (Development Environment)**

Estado de la regla	Texto de la regla
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The tools and supporting equipment used in product development, such as computer-aided software engineering (CASE) tools, simulators, compilers, and host computer systems.*
Formulación externa	**Si** Capacity = "yes" **y** Suitability = "yes" **y** Usability = "yes" **y** Familiarity = "no" **y** Reliability = "no" **y** System Support = "yes" **y** Deliverability = "no" **ENTONCES** Dev Sys-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Development System"
Nombre de la Regla	**R7 – Development System (Development Environment)**

Estado de la regla	Texto de la regla
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The planning, monitoring, and controlling of budgets and schedules; controlling factors involved in defining, implementing, and testing the product; the project manager's experience in software development, management, and the product domain; and the manager's expertise in dealing with external organizations including customers, senior management, matrix management, and other contractors.*
Formulación externa	**Si** Planning = "yes" **y** Project Organization = "yes" **y** Management Experience = "yes" **y** Program Interfaces = "no" **ENTONCES** Man Proc-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Management Process"
Nombre de la regla	**R8 – Management Process (Development Environment)**

Estado de la regla	Texto de la regla
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The methods, tools, and supporting equipment that will be used to manage and control the product development,*

	such as monitoring tools, personnel management, quality assurance, and configuration management.
Formulación externa	**Si** Monitoring = "yes" **y** Personnel Management = "yes" **y** Quality Assurance = "yes" **y** Configuration Management = "yes" **ENTONCES** Man Met-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Management Methods"
Nombre de la regla	**R9 – Management Methods (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The development environment class is concerned with the project environment in which a software product is engineered. The general environment within which the work will be performed, including the attitudes of people and the levels of cooperation, communication, and morale.*
Formulación externa	**Si** Quality Attitude = "no" **y** Cooperation = "no" **y** Communication = "no" **y** Morale = "no" **ENTONCES** Work Env-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Work Environment"
Nombre de la regla	**R10 – Work Environment (Development Environment)**

Estado de la regla	**Texto de la regla**
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but can still have major effects on its success. The external constraints imposed on schedule, staff, budget, or facilities.*
Formulación externa	**Si** Schedule = "no" **y** Staff = "no" **y** Budget = "no" **y** Facilities = "yes" **ENTONCES** Resources-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Resources"
Nombre de la regla	**R11 – Resources (Program Constraints)**

Estado de la regla	Texto de la regla
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but can still have major effects on its success. The terms and conditions of the project contract.*
Formulación externa	**Si** Type of Contract = "no" **y** Restrictions = "no" **y** Dependencies = "no" **ENTONCES** Contract-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Contract"
Nombre de la regla	**R12 – Contract (Program Constraints)**

Estado de la regla	Texto de la regla
Palabras del experto	*The program constraints class consists of the "externals" of the project—the factors that are outside the direct control of the project but can still have major effects on its success. The external interfaces to customers, other contractors, corporate management, and vendors.*
Formulación externa	**Si** Customer = "no" **y** Associate Contractors = "no" **y** Subcontractors = "no" **y** Prime Contractor = "no" **y** Corporate Management = "no" **y** Vendors = "yes" **y** Politics = "no" **ENTONCES** Interfaces-no-conclusion = "TRUE" Elemento de Clase Taxonómica = "Program Interfaces"
Nombre de la regla	**R13 – Program Interfaces (Program Constraints)**

Estado de la regla	Texto de la regla
Palabras del experto	*If there are no risks in the previous elements, then there are no risks in this system.*
Formulación externa	**Si** Requirements-no-conclusion = "TRUE" **y** Design-no-conclusion = "TRUE" **y** Test-no-conclusion = "TRUE" **y** Integration-no-conclusion = "TRUE" **y** Engineering-no-conclusion = "TRUE" **y** Dev Proc-no-conclusion = "TRUE"

	y Dev Sys-no-conclusion = “TRUE” **y** Man Proc-no-conclusion = “TRUE” **y** Man Met-no-conclusion = “TRUE” **y** Work Env-no-conclusion = “TRUE” **y** Resources-no-conclusion = “TRUE” **y** Contract-no-conclusion = “TRUE” **y** Interfaces-no-conclusion = “TRUE” **ENTONCES** Conclusion = “No risks were found”
Nombre de la regla	**R14 – No Risk**

En las formulaciones externas de cada una de las reglas citadas anteriormente se tomó el caso particular en que no se encuentran riesgos de ningún tipo para el desarrollo del software. Cualquier otra combinación en los valores de los elementos de las taxonomías dará como resultado un riesgo en el desarrollo, el cual se indicará como conclusión en la aplicación.

18. S.E. para la Identificación de Riesgos en el Desarrollo de Software

18.1 Análisis y diseño

18.1.1 Introducción

Mientras que en el clásico modelo cliente-servidor cada modificación al código del programa significa una nueva versión del lado del cliente, la propuesta de Java Server Pages (JSP) implica que solamente se modifique el código en un programa dentro del servidor y al instante ese cambio impactaría en los accesos de todos los usuarios. Es por esto, que se decidió utilizar JSP para el Sistema Experto propuesto en éste trabajo.

Por otra parte, en lo que respecta a los accesos en sí, el diseño a través del web permite de por sí que potencialmente no haya límites en la cantidad de usuarios conectados a la aplicación. En nuestro caso se tuvo que tener en cuenta la cantidad de instancias que se puedan levantar del motor de inferencia (ver mas adelante).

18.1.2 Arquitectura del sistema

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página y el código que provee de datos dinámicos a la página; el modelo es la lógica encerrada; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Profundizando en la Fig1.

Fig 1. Diagrama MVC de D.K.L.V Risk Identification System

El **Modelo** se encarga de la interacción entre el motor de inferencia y el código jsp. Obtiene las respuestas asociadas al cuestionario y las procesa.

La **Vista** presenta la información obtenida con el modelo de manera que el usuario la pueda visualizar. En nuestro caso es el resultado del cuestionario sobre los riesgos a enfrentar en el desarrollo software.

El **Controlador**, utilizando el patrón *filter*, recoge las respuestas y mediante un sistema satélite de NLP (procesamiento de lenguaje natural), envía las respuestas al modelo para que este interactué con el motor de inferencia y así poder lograr el resultado que se busca. Luego invoca a la página (de la vista) que corresponda para que la información sea presentada.

18.2 Configuración

18.2.1 Requerimientos de Hardware

- Un servidor, que formando parte de una red, provea servicios a los clientes. Los clientes en nuestra aplicación serán otras computadores conectadas a la red que, mediante el Internet Explorer u otro navegador, pueda acceder al sistema.

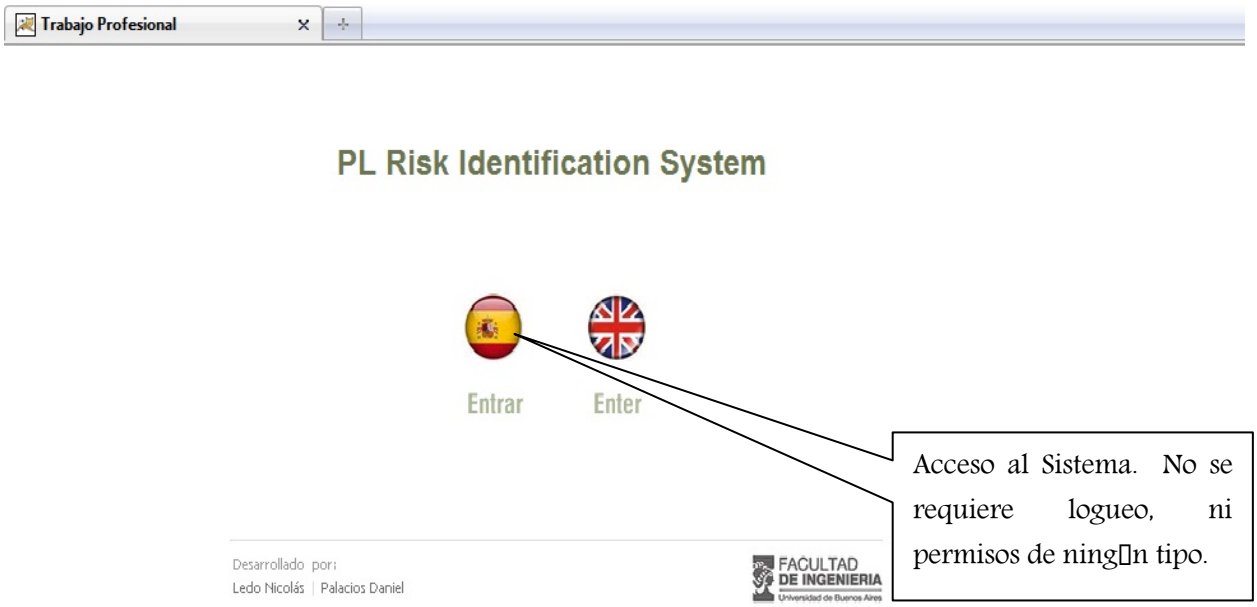
18.2.2 Requerimientos de Software

- Se recomienda el uso de Tomcat como servidor Web ya que soporta servlets y JSPs. También pueden ser otros servidores web como JBoss, JRun o JOnAs. Tomcat funciona en cualquier sistema operativo que disponga de la maquina virtual de java.
- Maquina virtual de java (JVM).
- Deben estar presentes en el sistema las siguientes librerías:
    - Freemarker 2.3.8
    - OGNL 2.6.11
    - Struts2-codebehind 2.0.12
    - Struts-core 2.0.12
    - Xwork 2.0.6
    - CLIPSJNI 0.2

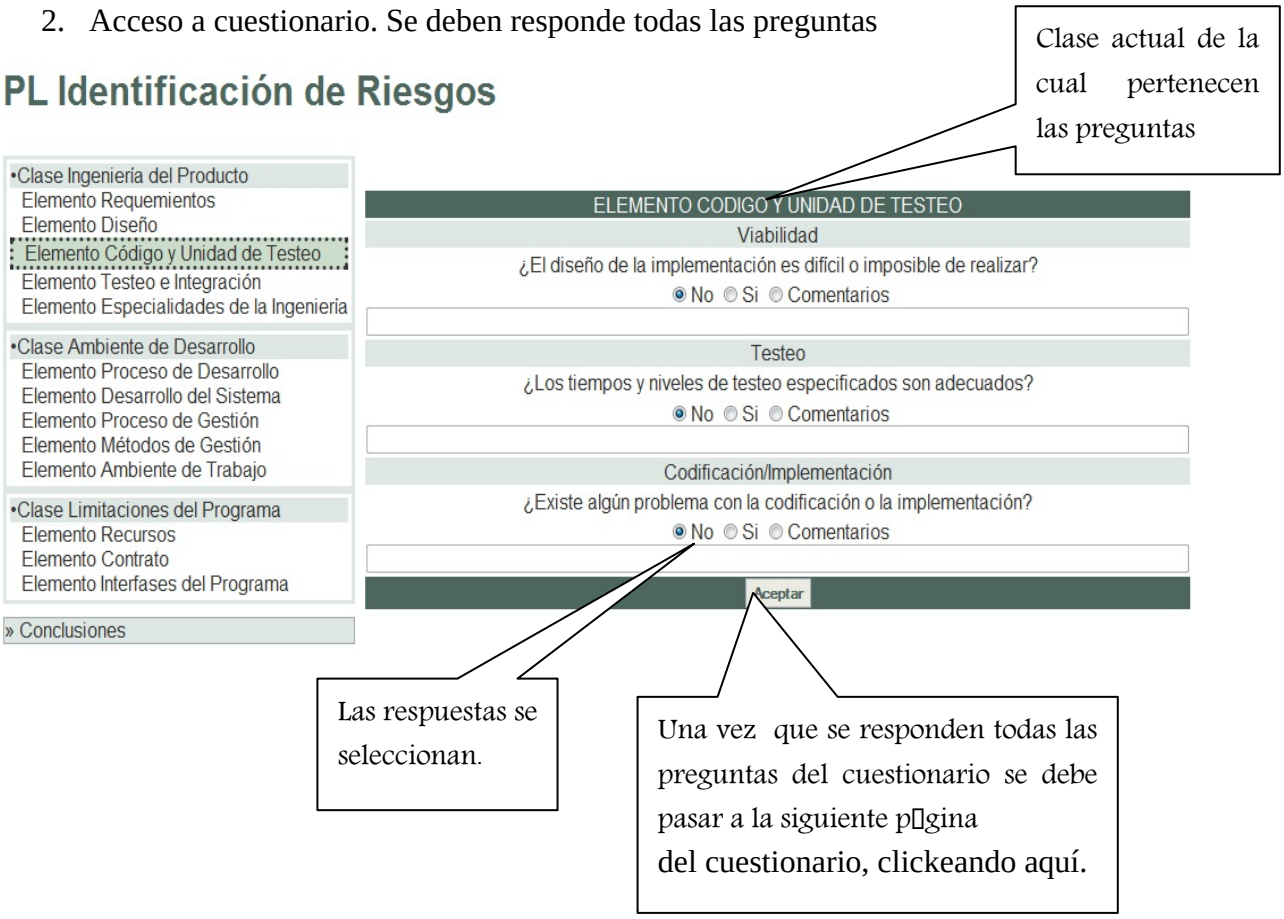
La librería CLIPSJNI consiste en un archivo dll y un archivo jar. Los mismos deben ser ubicados en la carpeta common/lib del Tomcat (o la carpeta equivalente a esta de otros servidores). Además debe incluirse a la misma en el PATH del sistema.

18.2.3 Ejecución – Modo de uso

1. Acceder a la página inicial. Se puede utilizar cualquier navegador. Se debe seleccionar el idioma a utilizar por el sistema.


Trabajo Profesional
PL Risk Identification System
Entrar
Enter
Desarrollado por:
Ledo Nicolás | Palacios Daniel
FACULTAD DE INGENIERIA
Universidad de Buenos Aires
Acceso al Sistema. No se requiere logueo, ni permisos de ningún tipo.

2. Acceso a cuestionario. Se deben responde todas las preguntas

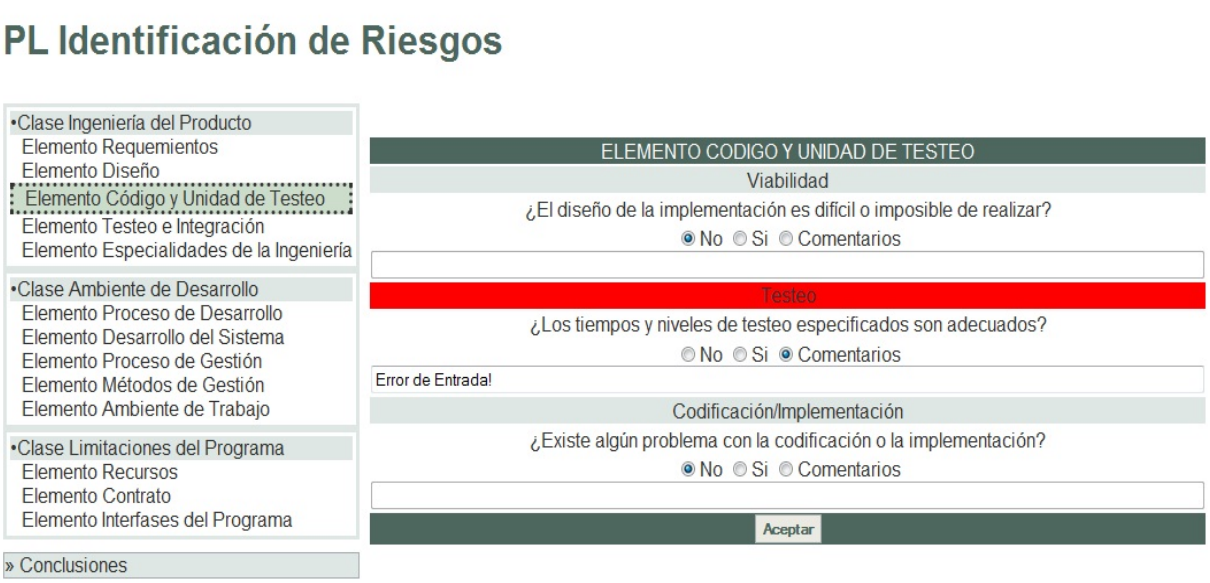

PL Identificación de Riesgos
•Clase Ingeniería del Producto
Elemento Requemientos
Elemento Diseño
Elemento Código y Unidad de Testeo
Elemento Testeo e Integración
Elemento Especialidades de la Ingeniería
•Clase Ambiente de Desarrollo
Elemento Proceso de Desarrollo
Elemento Desarrollo del Sistema
Elemento Proceso de Gestión
Elemento Métodos de Gestión
Elemento Ambiente de Trabajo
•Clase Limitaciones del Programa
Elemento Recursos
Elemento Contrato
Elemento Interfases del Programa
» Conclusiones
ELEMENTO CODIGO Y UNIDAD DE TESTEO
Viabilidad
¿El diseño de la implementación es difícil o imposible de realizar?
No Si Comentarios
Testeo
¿Los tiempos y niveles de testeo especificados son adecuados?
No Si Comentarios
Codificación/Implementación
¿Existe algún problema con la codificación o la implementación?
No Si Comentarios
Aceptar
Clase actual de la cual pertenecen las preguntas
Las respuestas se seleccionan.
Una vez que se responden todas las preguntas del cuestionario se debe pasar a la siguiente p□gina del cuestionario, clickeando aquí.

3. Terminadas de responder todo el cuestionario, el informe saldrá en este formato.

Esta salida es en el caso de que se detecten riesgos.

En caso de no detectar ningún riesgo, la salida del sistema será la siguiente:

Existe la posibilidad que el usuario ingrese una respuesta que este fuera del dominio definido para la aplicación. Es decir, que la respuesta no sea coherente, o bien, su formulación no posea un significado acorde a lo que se le preguntó. En este caso, cuando el usuario presione el botón “Aceptar” luego de haber llenado las preguntas de alguna de las secciones, ocurrirá lo que se muestra a continuación:



Si se observa la figura se puede ver claramente como ante una respuesta incoherente el sistema resalta en rojo el atributo cuya respuesta fue erróneamente contestada. En este caso en particular, el atributo “Testeo” tuvo una respuesta incorrecta y para que el usuario pueda continuar contestando las preguntas del sistema experto deberá reformular su respuesta.

19. Anexo II. Procesamiento de lenguaje natural

19.1 Introducción - NLP (Natural Language Processing)

El Procesamiento de Lenguajes Naturales (abreviado PLN, o NLP del idioma inglés Natural Language Processing) es una subdisciplina de la Inteligencia Artificial y la rama ingenieril de la lingüística computacional. El PLN se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas o entre personas y máquinas por medio de lenguajes naturales. El PLN no trata de la comunicación por medio de lenguajes naturales de una forma abstracta, sino de diseñar mecanismos para comunicarse que sean eficaces computacionalmente que se puedan realizar por medio de programas que ejecuten o simulen la comunicación. Los modelos aplicados se enfocan no sólo a la comprensión del lenguaje de por sí, sino a aspectos generales cognitivos humanos y a la organización de la memoria. El lenguaje natural sirve sólo de medio para estudiar estos fenómenos.

Dificultades en el procesamiento de lenguajes naturales

- Ambigüedad: surge cuando una expresión hablada o escrita posee más de un significado o interpretación.
- Recepción imperfecta de datos: acentos extranjeros, regionalismos o dificultades en la producción del habla, errores de mecanografiado o expresiones no gramaticales, etc.
- Análisis sintáctico y semántico: analizar la forma y el sentido de una expresión.

19.2 Soluciones e implementación propuesta

A lo largo del desarrollo del sistema nos enfrentamos con varios inconvenientes. Uno de ellos fue desarrollar un procesador de lenguaje natural que se adaptara a nuestras necesidades. Hoy en día existen varias herramientas open source como es el caso de Open NLP que proveen al usuario muchas facilidades a la hora de resolver un problema tan grande como es la interpretación de un lenguaje natural. Como contrapartida son herramientas muy complejas que necesitan de una adaptación muy grande para lo que nosotros necesitábamos.

¿Cuáles eran nuestras necesidades frente a este problema?

- PLN de lenguaje **inglés** dado que nuestro sistema experto fue realizado en este idioma, además de en español.
- Reducir las respuestas del usuario del sistema experto a expresiones simples del tipo **"yes" o "no"** dadas las necesidades de nuestro motor de inferencia.
- Simplicidad y velocidad de procesamiento dado que es un sistema que puede tener varias aplicaciones realizando consultas concurrentemente al motor de inferencia.
- Encontrar una manera de acotar el dominio de respuestas posibles del usuario dada la diversidad y complejidad que pueden tener las mismas.

¿Cuál fue nuestra solución a este conjunto de problemas?
Después de realizar un análisis exhaustivo de la situación encontramos una solución a todos los problemas que habían ido surgiendo.
Lo que hicimos fue definir un dominio de términos (en idioma inglés) que denotarán aceptación y negación. El mismo fue definido en base a las preguntas realizadas por el sistema experto al usuario y el sentido común.
El módulo denominado como **NLP** recibe como entrada la frase ingresada por el usuario, la parsea y luego la analiza. Del análisis se pueden obtener 3 respuestas posibles:

1. La frase posee connotación negativa por consiguiente es análoga a un "no".
2. La frase posee connotación positiva por consiguiente es análoga a un "yes".
3. La frase no se encuentra bien formulada o carece de la información necesaria, con lo cual su estado es "indefinido" y el usuario, eventualmente deberá especificar su respuesta o bien reformularla.

El análisis de la frase que realiza NLP esta basado en un análisis probabilístico y estadístico en base a la cantidad de "matcheos" de cada uno de los términos que componen la frase contra el dominio de palabras de connotación negativa y positiva que definimos. De esta forma simple logramos una velocidad de respuesta mucho mayor sin recurrir a soluciones de alta complejidad que no solo dificultan el procesamiento sino que no se adaptan a nuestras necesidades.

Como conclusión podemos observar que este módulo de alguna manera tiene relación directa con la interfaz (la vista) y el motor de inferencia que es el módulo a más bajo nivel. Cada respuesta del usuario (luego de ser analizada) será enviada al motor de inferencia que luego de recopilar todas las respuestas del usuario llegará a la conclusión definitiva.

Creemos que la solución propuesta a este problema fue muy eficiente, simple y práctica y se adaptó excelentemente al proyecto realizado.

Anexo III. Herramientas utilizadas y aclaraciones

I.D.E.: Eclipse SDK, version 3.3.1.

Tomcat 5.5. Para hacer más sencillo su uso, se instaló el plugin de Tomcat para el Eclipse. El mismo se puede descargar desde: http://apache.localhost.net.ar/tomcat/tomcat-5/v5.5.25/bin/apache-tomcat-5.5.25.zip. Para instalarlo lo que se debe hacer es descomprimir su contenido en la carpeta plugins de eclipse.

Con respecto a la configuración del Tomcat dentro de Eclipse (para esto ya debe estar instalado el plugin):

- Menu Window -> Preferences.
- Solapa Tomcat.
- En Tomcat Version, elegir 5.x.
- En Tomcat Home, poner la ruta de la carpeta del Tomcat.
- En Context Declaration Mode, elegir Context Files.
- En la sub-solapa Advanced, en Tomcat base, poner la ruta de la carpeta del Tomcat.

Para poder correr la aplicación se debe crear un proyecto Tomcat y volcar el contenido de las subcarpetas **js** y **jsp** de la carpeta que se encuentra en el cd bajo el nombre **TrabajoProfesional** en la carpeta del proyecto creado.

Para correr la aplicación debe ejecutarse el Tomcat y luego desde el browser ingresar la siguiente URL: http://localhost:8080/TrabajoProfesional/jsp/index.jsp.

21. Anexo IV. Usability Manual

21.1 Configuration

21.1.1 Hardware Requirements

- A server, forming part of a network that provides services to clients. Customers in our application would be other computers connected to the network and through the Internet Explorer or other browser can access the system.

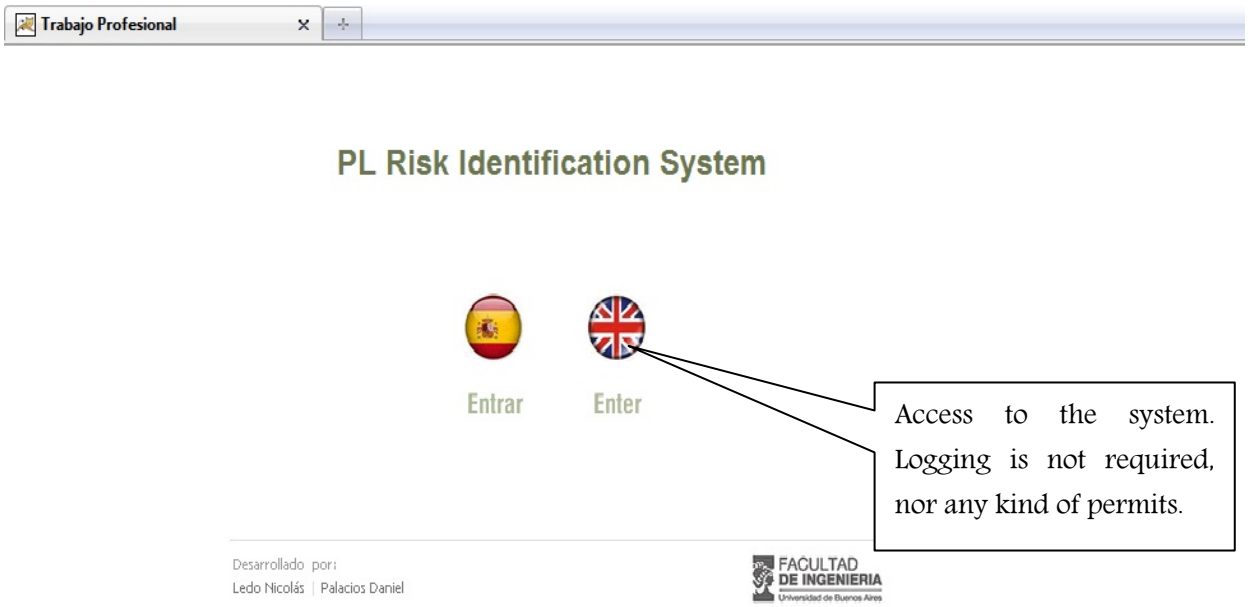
21.1.2 Software Requirements

- We recommend using Tomcat as Web Server because it supports Servlets and JSPs. You may use other Web Servers like: JBoss, JRun or Jonas. Tomcat runs on any operating system that has the Java Virtual Machine.
- Java Virtual Machine (JVM).
- Must be present in the system the following libraries:
    - Freemarker 2.3.8
    - OGNL 2.6.11
    - Struts2-codebehind 2.0.12
    - Struts-core 2.0.12
    - Xwork 2.0.6
    - CLIPSJNI 0.2

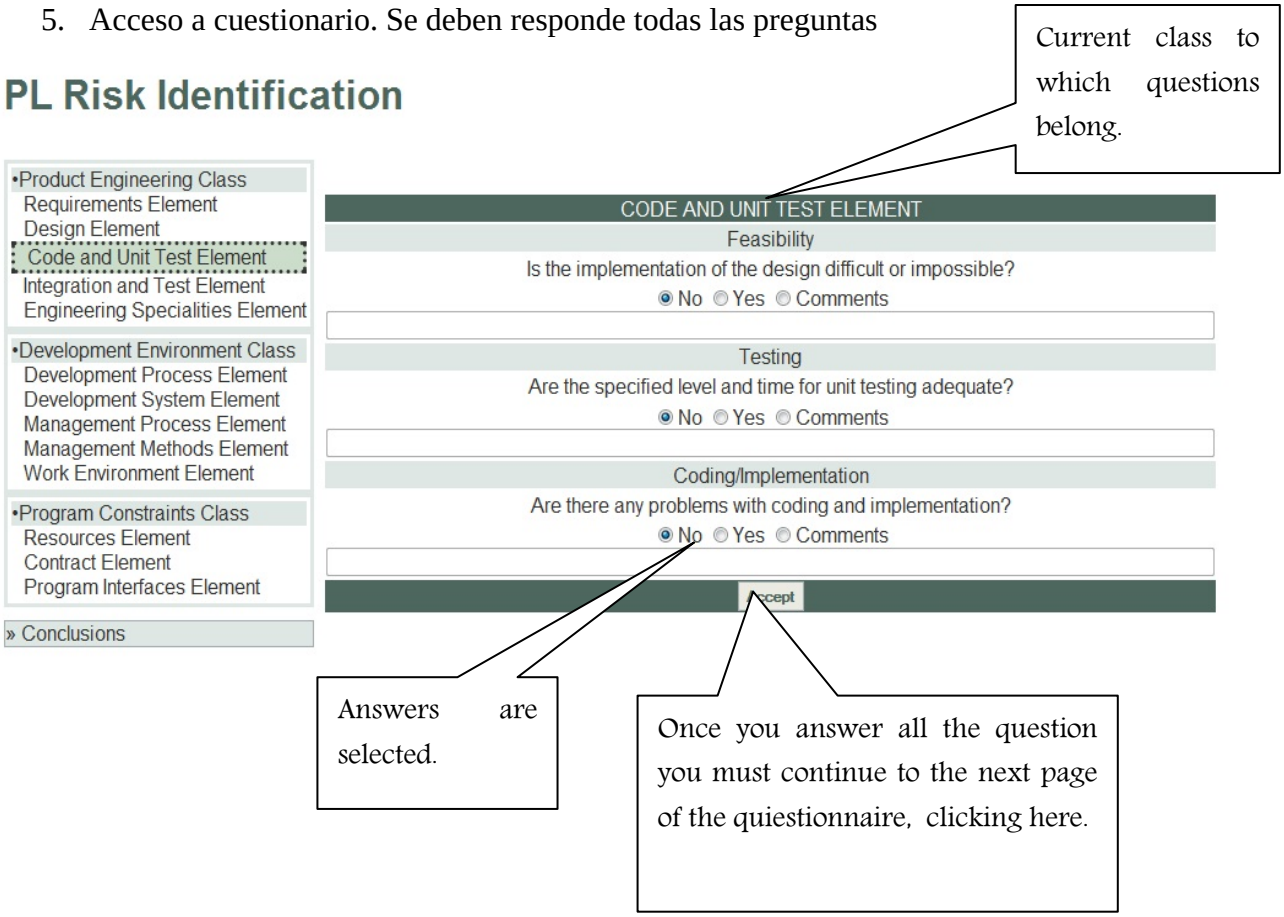
The CLIPSJNI library is a dll file and a jar file. They should be located in the folder common/lib of Tomcat folder (or the equivalent to that of other servers). In addition it must be included in the PATH system.

21.2 Execution – Usage (English Version)

4. Access the Main Page. You can use any browser. You must select the language that will be used by the system.

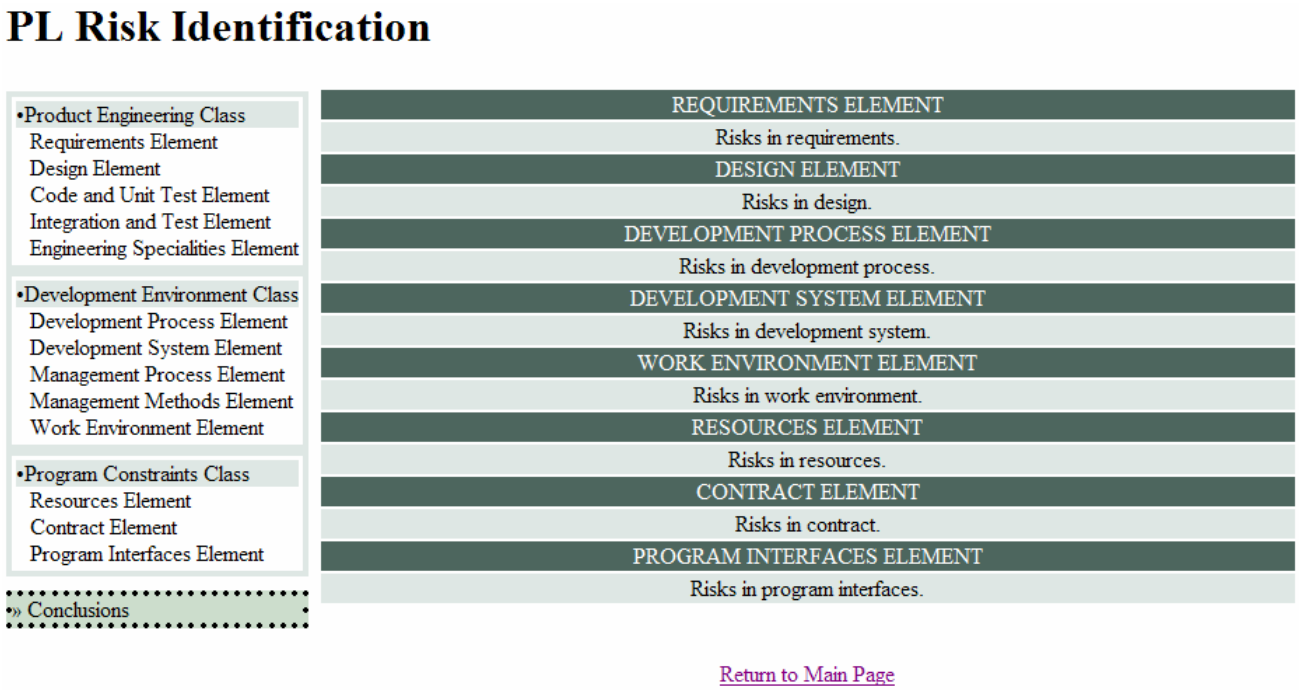


5. Acceso a cuestionario. Se deben responde todas las preguntas

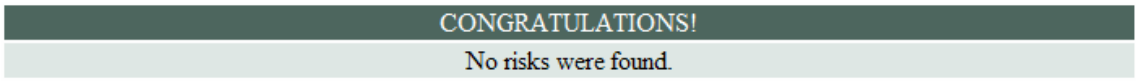


6. Once the questionnaire is answered, the report will appear in the following format:

This is the output in case risks are identified.

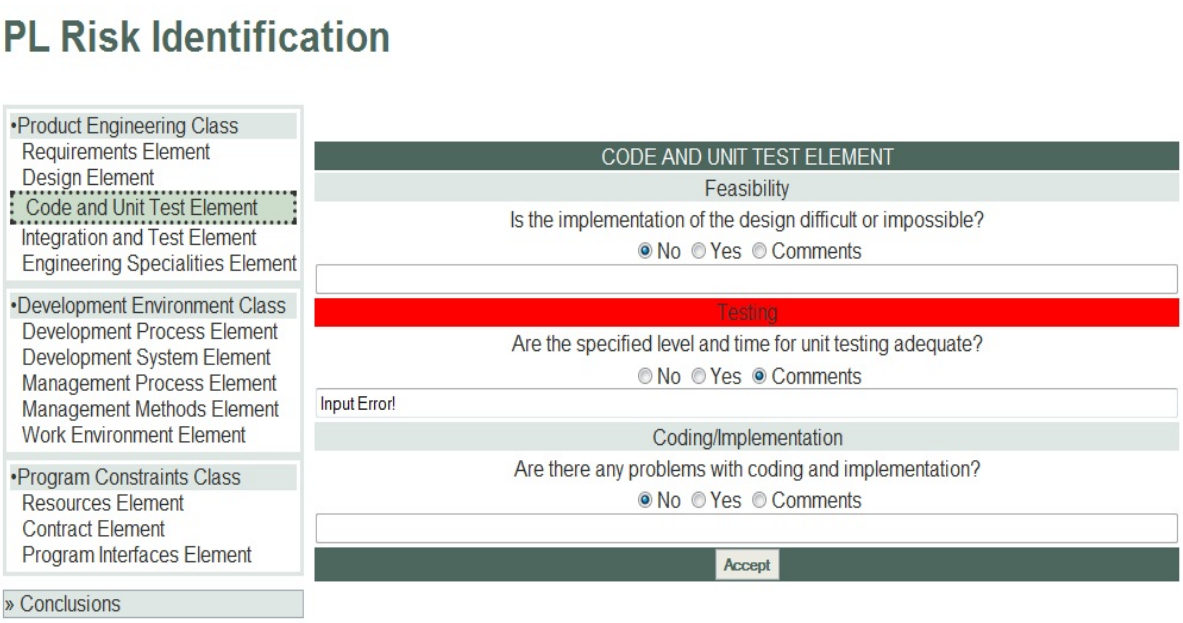


If any risks are detected, the system output will be the following:



There is a possibility that the user enters an answer that is outside the domain defined by the application. This means that the answer is inconsistent, or its formulation does not have a consistent meaning to what was asked.

In this case, when the user clicks the "Accept" button after filling the questions of one of the sections, the output will, be the following:



As you can see clearly in the image when a question is wrongly answered the system highlights in red the attribute whose answer was incoherent. In this particular case, the attribute "Testing" was the wrongly answered and the user can continue answering the questions of the expert system only if he answers the question again correctly.